Improve your data science with 'Efficient R Programming'

Brandon C. Loudermilk

May 15, 2017

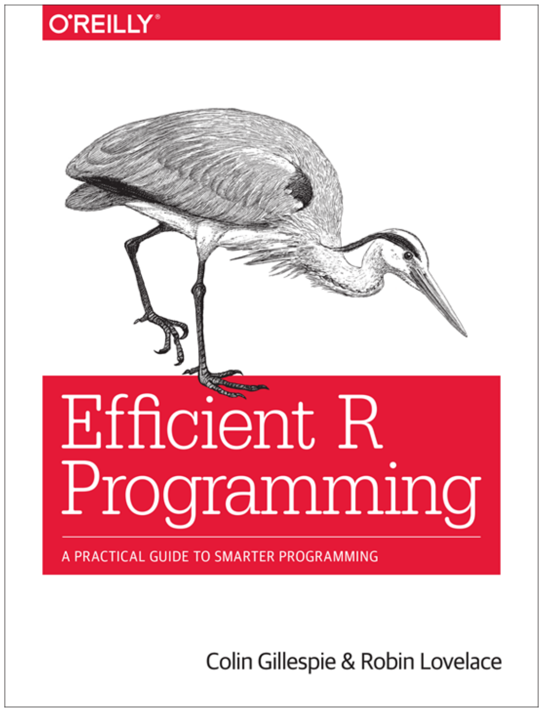

O'REILLY
Efficient R Programming
A PRACTICAL GUIDE TO SMARTER PROGRAMMING
Colin Gillespie & Robin Lovelace

Figure 1:

In the crowded market space of data science and the R language books, Lovelace and Gillespie's Efficient R Programming (2016) stands out from the crowd. Over the course ten comprehensive chapters, the authors address the primary tenets of writing and developing efficient R programs. Unless you happen to be a member of the R core development team, you will find this book useful whether you are a novice R programmer or an established data scientist and engineer. This book is chock full of useful tips and techniques that will help you improve the efficiency of your R programs as well as the efficiency of your development processes. Although I have been using R daily (and nearly exclusively) for the past 4+ years, every chapter of this book provided me with new insights into how to improve my R code while helping solidify my understanding of previously learned techniques. Each chapter of **Efficient R Programming** is devoted to a single topic, each of which includes a "top five tips" list, covers numerous packages/techniques, and contains useful exercises and problem sets for consolidating key insights.

In **Chapter 1. Introduction**, the authors orient the audience to the key characteristics of R that bear on its efficiency compared to other programming languages. Importantly, the authors address R efficiency not just in the expected sense of algorithmic speed and complexity but broaden its scope to include programmer productivity and how it relates to programming idioms, IDEs, coding conventions, and community support – all things that can improve the efficiency of writing and maintaining code. This is doubly important for a language like R which is notoriously flexible in its ability to solve problems in multiple ways. The first chapter concludes by introducing the reader to two valuable packages: (1) microbenchmark – an accurate benchmarking tool with nanosecond precision; and (2) profvis – a handy tool for profiling larger chunks of code. These two packages are repeatedly used throughout the remainder of the book to illustrate key concepts and highlight efficient techniques.

In **Chapter 2. Efficient Setup** the reader is introduced to techniques for setting up a development environment that facilitates efficient workflow. Here the authors cover choices in operating system, R version, R start-up, alternative R interpreters, and how to maintain up-to-date packages with tools like packrat and installr. I found their overview of the R startup process particularly useful, as the authors taught me how to modify my **.Renviron** and **.Rprofile** files to cache external API keys and customize my R environment, for example by adding alias shortcuts to commonly used functions. The chapter concludes by discussing how to setup and customize the RStudio environment (e.g., modifying code editing preference, editing keyboard shortcuts, and turning off restore **.Rdata** to help prevent bugs) which can greatly improve individual efficiency.

Chapter 3. Efficient Programming introduces the reader to efficient programming by discussing "big picture" programming techniques and how they relate to the R language. This chapter will most likely be beneficial to established programmers who are new to R as well as data scientists and analysts who have limited exposure to programming in a production environment. In this chapter the authors introduce the "golden rule of R programming" before delving into the usual suspects of inefficient R code. Usefully, the book illustrates multiple ways of performing the same task (e.g., data selection) with different code snippets and highlights the performance differences through benchmarked results. Here we learn about the pitfalls of growing vectors, the benefits of vectorization, and the proper use of factors. The chapter wraps up with the requisite overview of the apply function family, before discussing the use of variable caching (package memoise) and byte compilation as important techniques in writing fast, responsive R code.

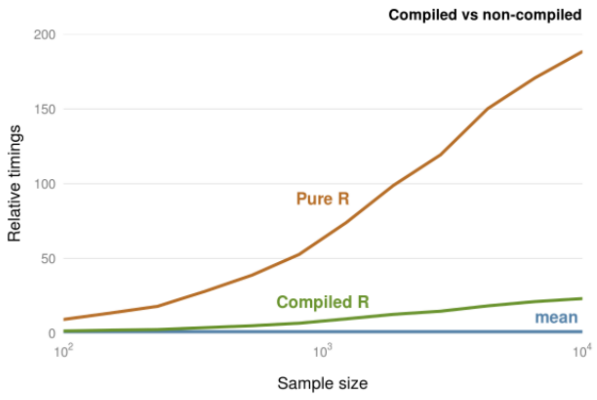


Figure 3-4. Comparsion of mean functions

Figure 2:

Chapter 4. Efficient Workflow will be of primary use to junior level programmers, analysts, and project managers who haven't had enough time or practice to develop their own efficient workflows. This chapter discusses the importance of project planning, audience, and scope before delving into common tools that facilitate project management. In my opinion one of best aspects of R is the huge, maddeningly broad number of packages that are available on CRAN and GitHub. The authors provide useful advice and techniques for searching and identifying the packages that will be of most use to your project. A brief discussion on the use of R Markdown and knitr concludes this chapter.

Chapter 5. Efficient Input/Output is devoted to efficient read/write operations. Anybody who has ever struggled with loading a big file into R for analysis will appreciate this discussion and the packages covered in this chapter. The rio package, which can handle a wide variety of common data file types, provides a useful starting point for exploratory work on a new project. Other packages that are discussed (readr, data.table) provide more efficient I/O than those in base R. The chapter ends with a discussion of two new file formats and associated packages (feather and RProtoBuf) that can be used for cross-language fast, efficient serialized data I/O.

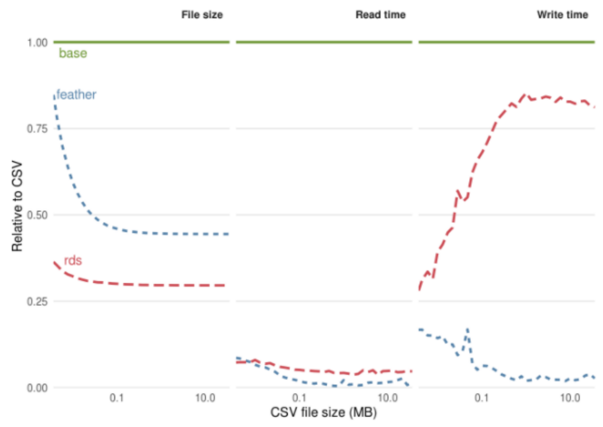


Figure 5-2. Comparison of the performance of binary formats for reading and writing datasets with 20 columns with the plain-text format CSV; the functions used to read the files were read.csv(), readRDS(), and feather::read_feather(), respectively. The functions used to write the files were write.csv(), saveRDS(), and feather::write_feather().

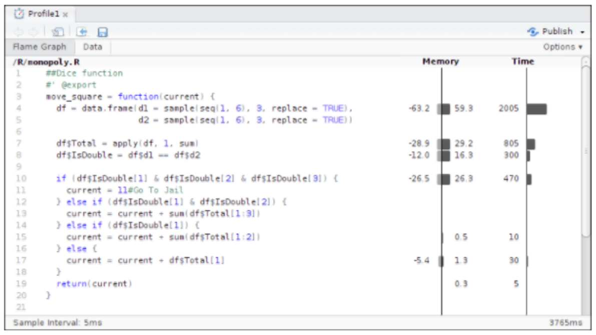
Figure 3:

Chapter 6. Efficient Data Carpentry introduces what are in my opinion, the most useful R tools for *data munging* – what Lovelace and Gillespie prefer to call by the more admirable term "data carpentry." This chapter could more aptly be titled the "Tidyverse" or the "Hadleyverse", for most of the tools discussed in this chapter were developed by prolific R package writer, Hadley Wickham. Sections of the chapter are devoted to each of the primary packages of the tidyverse: tibble a more useful and user-friendly data.frame; tidyr for reshaping data between short and long forms; stringr provides a consistent API over obtuse regex functions; dplyr for efficient data processing including filtering, sorting, mutating, joining, and summarizing; and of course magrittr for piping all these operations together with %>%. A brief section on package data.table rounds out the discussion on efficient data carpentry.

Chapter 7. Efficient Optimization begins with the requisite optimization quote by computer scientist Donald Knuth:

> The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil (or at least most of it) in programming.

In this chapter the authors introduce profvis and they illustrate the utility of this package by showing how it can be used to identify bottlenecks in a Monte Carlo simulation of a Monopoly game. The authors next examine alternative methods in base R that can be used for greater efficiency. These include discussion of `if()` vs. `ifelse()`; sorting operations, AND (`&`) and OR (`|`) vs. `&&` and `||`, row/column operations, and sparse matrices. The authors then apply these tricks to the Monopoly code where they show a 20-fold decrease in run-time. The chapter concludes with discussion and examples of parallelization and the use of Rcpp as an R interface to underlying fast and efficient C++ code.



```
/R/monopoly.R                                                      Memory        Time
##Dice function
#' @export
move_square = function(current) {
  df = data.frame(d1 = sample(seq(1, 6), 3, replace = TRUE),    -63.2   59.3   2005
                  d2 = sample(seq(1, 6), 3, replace = TRUE))

  df$Total = apply(df, 1, sum)                                 -28.9   29.2    805
  df$IsDouble = df$d1 == df$d2                                 -12.0   16.3    300

  if (df$IsDouble[1] & df$IsDouble[2] & df$IsDouble[3]) {      -26.5   26.3    470
    current = 11#Go To Jail
  } else if (df$IsDouble[1] & df$IsDouble[2]) {
    current = current + sum(df$Total[1:3])
  } else if (df$IsDouble[1]) {
    current = current + sum(df$Total[1:2])                             0.5     10
  } else {
    current = current + df$Total[1]                             -5.4    1.3     30
  }
  return(current)                                                      0.3      5
}

Sample Interval: 5ms                                                           3765ms
```

Figure 7-2. Code profiling for simulating the game of Monopoly

Figure 4:

I found the chapter **Efficient Hardware** to be the least useful in the book (spoiler alert: add more RAM or migrate to cloud-based services), though the chapter on **Efficient Collaboration** will be particularly useful for novice data scientists lacking real-world experience developing data artifacts and production applications in a distributed, collaborative environment. In this chapter, the authors discuss the importance of coding style, code comments, version control, and code review. The final chapter **Efficient Learning**, will find appreciative readers among those just getting started with R (and if this describes you, I would suggest that you start with this chapter first!). Here the authors discuss using and navigating R's excellent internal help utility as well as the importance of vignettes and source code in learning/understanding. After briefly introducing swirl, the book concludes with a discussion of online resources, including Stack Overflow; the authors thankfully provide the newbie with important information on how to ask the right questions and the importance of providing a great R reproducible example.

In summary, Lovelace and Gillespie's **Efficient R Programming** does an admirable job of illustrating the key techniques and packages for writing efficient R programs. The book will appeal to a wide audience from advanced R programmers to those just starting out. In my opinion, the book hits that pragmatic sweet spot between breadth and depth and it usefully contains links to external resources for those wishing to delve deeper into a specific topic. After reading this book, I immediately went to work refactoring a Shiny dashboard application I am developing and several internal R packages I maintain for our data science team. In a matter of a few short hours I witnessed a 5 to 10-fold performance increase in these applications just by implementing a couple of new techniques. I was particularly impressed with the greatly improved end-user performance and the ease with which I implemented intelligent caching with the memoise package for a consumer decision tree application I am developing. If you care deeply about writing beautiful, clean, efficient code and bringing your data science to the next level, I highly recommend adding Efficient R Programming to your arsenal.

The book is published by O'Reilly Media and is available online at the authors' website as well as through

Safari.