Malte Grosser, Henning Bumann & Hadley Wickham

Advanced R Solutions

To

Contents

1

Preface

Welcome to Advanced R Solutions!

This book provides worked-out solutions to the exercises given in Advanced R (2nd Edition, Wickham, 2019) and reflects our efforts to understand and acquire its content.

Advanced R covers R and programming. It presents the specific characteristics of the R language to programmers and helps R users to improve their understanding of general programming concepts.

When we came across Advanced R, it gave us a more thorough understanding of the R code we worked with daily and helped us to see the underlying principles more clearly. The content helped us to get a more complete picture of R's programming landscape.

We soon re-discovered that reading about programming is not enough and that it helps greatly to open the editor and write some code along the way. The clear structure of Advanced R and the exercises given provided a great starting point for this.

We think of this book as a solutions manual, which intends to supplement and support your own study of the R language through Advanced R. We hope that it will help you to stay on track and allow you to check your understanding and progress along the way. The solutions may also give you another perspective on some of the presented material.

1.1 How this book came to be

The solutions in this book are written from our personal perspective and current level of understanding. We both come from mathematics and statistics backgrounds preparing us more carefully for data analysis than for programming. So, we were R users first and improved as programmers in our day jobs and through working on projects like this one.

By taking advantage of the `{bookdown}` package to structure our process we created this book almost as a by-product. While the initial progress was fun and motivating, providing solutions to actually all of the 284 exercises took slightly longer than originally expected (and Hadley's rewrite of Advanced R halfway in between didn't really make the journey shorter).

As the project matured, we strived to provide solutions as clean, straightforward, and expressive as possible. As well written code is often more expressive than lengthy explanations, many of the solutions are rather code-heavy. The written explanations aim to fill in context and motivation, discuss important implementation details or relate to the practical work of being an R programmer.

Hadley Wickham wrote Advanced R and created the exercises which form the substructure of this book. We took the task to solve them as correctly and idiomatically as possible. When we finished a chapter, we asked Hadley to review it. His feedback included many comments (which we then had to resolve), corrections and suggestions, as well as a few complete solutions. We repeated this process until each exercise was reviewed and approved. As a result, we feel pretty good about the quality of the solutions in the book. However, any remaining mistakes or inconsistencies are certainly on us.

1.2 How to use this book

Since this book builds so heavily on Advanced R, we think it should be read together with the textbook, either as a hardcopy or the online version (`https://adv-r.hadley.nz`). Working on an exercise first by yourself should in general give you the biggest benefit.

It may be a good idea to start with the parts of Advanced R that are most relevant to your work and interest. You can certainly read the book cover to cover, but we think that you don't have to, though it's probably a good idea to start with the foundations part.

Of the more difficult exercises in the book, only a few were completed in one go. Often we had to reread the question or look up the related content in Advanced R and started by writing a few first lines of code or consulted the documentation. Reading the source code (preferably with syntax highlighting) and searching the web were typically quite helpful.

To support your study, you may also be interested in the R4DS Advanced R book club (`https://GitHub.com/r4ds/bookclub-Advanced_R`), where groups of readers regularly discuss a different chapter of Advanced R.

In case you want to do more or have a special interest in the mechanics of base R, you may be interested in checking out the first edition of Advanced R (`http://adv-r.had.co.nz/`). Some additional solutions related to that edition can be found at `https://advanced-r-solutions-ed1.netlify.app/`.

There is one recommendation from Advanced R that we'd like to echo: reading source code can be tremendously helpful in developing your programming skill! For example, you can just head to GitHub and start looking into the source code of packages you love and developers you admire. While reading, it's not necessary to understand every line of code right away. Keeping this a regular practice (for a while) will expose you to many new ideas, patterns, design choices and also expand your R vocabulary.

We don't necessarily apply many of the concepts taught in Advanced R in daily programming and that's okay! But we hope that the code we write has become more robust, expressive, and readable and it's actually quite easy to see the progress, when we take a look at the earlier drafts of our own code.

1.3 Acknowledgements

Many open source projects are the result of the work of a lot of people; so is this. We would like to explicitly mention and thank everybody who contributed solutions, raised questions, or helped to fix spelling and grammar to improve this work.

@3zhang, Jun Cai (@arashHaratian), Leon Kim (@BetweenTwoTests), @caijun, PJ (@Charles926), Safouane Chergui (@chsafouane), Corrado Lanera (@CorradoLanera), @davidblitz, Zhuoer Dong (@dongzhuoer), @Elucidase, Fabian Scheipl (@fabian-s), @its-gazza, Jorge Aranda (@jorgearanda), @lotgon, @MajoroMask, Maya Gans (@MayaGans), Øystein Sørensen (@osorensen), Peter Hurford (@peterhurford), @philyoun, Arash (@pieterjanvc), Robert Krzyzanowski (@robertzk), Tanner Stauss (@tmstauss), Anh N Tran (@trannhatanh89), Yihui Xie (@yihui).

Tobias Stalder (`@toeb18` (`https://twitter.com/toeb18`)) designed the beautiful cover, which visualizes the structure of Advanced R and its exercises.

Thanks to CRC Press for the interest in the project and our editors Rob Calver and Vaishali Singh for their patience and support in making this book a reality.

Thanks to our managers and companies for granting us some flexibility with our work schedules and generally supporting the completion of this project.

1.4 Conventions

A brief overview of conventions we followed and decisions we made.

- Some chapters and sections in Advanced R do not contain exercises. In our book you will see that we skipped these chapters and sections. This decision introduces some gaps in the numbering, but we feel that keeping the numbers in sync with those of Advanced R will provide the most practical value.
- We strived to follow mostly the tidyverse style guide (`https://style.tidyverse.org/`) (using the `{styler}` package (**?**) made this a bit easier).
- Each chapter of this book was rendered in a separate R session via the `{bookdown}` package. We configured this process to initially
  - set `` `%>%` <- magrittr::`%>%` `` to unlock the pipe operator without specifically loading the `{magrittr}` package (**?**) every time,
  - set a random seed (`1014`) to improve reproducibility (similar as in Advanced R), and
  - define a few `{ggplot2}` and `{knitr}` options.

  You can check out the exact code (`https://GitHub.com/Tazinho/Advanced-R-Solutions/blob/main/common.R`) on GitHub.
- We chose to keep the code in this book as self-contained as possible.
  - The packages used are usually loaded in the beginning of each chapter.
  - We repeated all code from Advanced R that is necessary to work on an exercise but not explicitly part of the exercise. When some longer code passages (from Advanced R) were omitted this is explicitly stated in the solution.
- The printed version of the book was rendered with R version 4.0.3 (2020-10-10) and the most recent available package versions as of December 2020.
- Emoji images in the printed book come from the open-licensed Twitter Emoji (`https://github.com/twitter/twemoji`).
- Benchmarks are computed when the book is rendered. While this improves reproducibility, the exact results will depend on the system creating the document.

1.5 Closing remarks

We are so happy to finish this exciting project, that in fact neither of us really had the time for. We probably wouldn't have made it to the finish line if we hadn't worked on it together.

Collaboration is powerful and it's fun to build and share. The various backgrounds represented in the R community generally make this exchange much more interesting and meaningful. Much of this success is possible because R is free software. At least in theory everyone can contribute, and no one can take away your freedom to do so.

The automated systems we build using these tools are not neutral and the rapid adoption of data-driven processes in business and technology does clearly affect our everyday lives and societies. It's important that everyone has a fair say in the discussions about these systems and participate in their design. Against this background we chose to donate half of our royalties from this book to `https://rladies.org/`, an organization empowering female R users.

Thank you for your interest in this project and we hope the solutions will be of value to you.

See you around!

Malte Grosser `@malte_grosser` (`https://twitter.com/malte_grosser`)

Henning Bumann `@henningsway` (`https://twitter.com/henningsway`)

Part I

Foundations

2

Names and values

Prerequisites

We will use the `{lobstr}` package to help answer questions regarding the internal representation of R objects.

```
library(lobstr)
```

2.2 Binding basics

Q1: Explain the relationship between `a`, `b`, `c`, and `d` in the following code:

```
a <- 1:10
b <- a
c <- b
d <- 1:10
```

A: `a`, `b`, and `c` point to the same object (with the same address in memory). This object has the value `1:10`. `d` points to a different object with the same value.

```
list_of_names <- list(a, b, c, d)
obj_addrs(list_of_names)
#> [1] "0x196fa590" "0x196fa590" "0x196fa590" "0x197de790"
```

Q2: The following code accesses the mean function in multiple ways. Do they all point to the same underlying function object? Verify this with `lobstr::obj_addr()`.

```
mean
base::mean
get("mean")
evalq(mean)
match.fun("mean")
```

A: Yes, they point to the same object. We confirm this by inspecting the address of the underlying function object.

```
mean_functions <- list(
  mean,
  base::mean,
  get("mean"),
  evalq(mean),
  match.fun("mean")
)

unique(obj_addrs(mean_functions))
#> [1] "0x138446f0"
```

Q3: By default, base R data import functions, like `read.csv()`, will automatically convert non-syntactic names to syntactic ones. Why might this be problematic? What option allows you to suppress this behaviour?

A: Column names are often data, and the underlying `make.names()` transformation is non-invertible, so the default behaviour corrupts data. To avoid this, set `check.names = FALSE`.

Q4: What rules does `make.names()` use to convert non-syntactic names into syntactic ones?

A: A valid name may contain letters, numbers, dots, and underscores (`"_"` are allowed since R version 1.9.0). It must start with a letter or a dot. If it starts with a dot, it must not be followed by a number.

Three main mechanisms ensure syntactically valid names (see `?make.names`):

1. Names that do not start with a letter or a dot will be prepended with an `"X"`.

   ```
   make.names("")    # prepending "x"
   #> [1] "X"
   ```

The same holds for names that begin with a dot followed by a number.

```
make.names(".1")  # prepending "X"
#> [1] "X.1"
```

2. Additionally, non-valid characters are replaced by a dot.

```
make.names("@")          # prepending "X" + "." replacement
#> [1] "X."
make.names("  ")         # prepending "X" + ".." replacement
#> [1] "X.."
make.names("non-valid")  # "." replacement
#> [1] "non.valid"
```

3. Reserved R keywords (see `?reserved`) are suffixed by a dot.

```
make.names("if")  # "." suffix
#> [1] "if."
```

Interestingly, some of these transformations are influenced by the current locale. From `?make.names`:

> The definition of a letter depends on the current locale, but only ASCII digits are considered to be digits.

Q5: I slightly simplified the rules that govern syntactic names. Why is `.123e1` not a syntactic name? Read `?make.names` for the full details.

A: `.123e1` is not a syntactic name, because it starts with one dot which is followed by a number. This makes it a double, `1.23`.

2.3 Copy-on-modify

Q1: Why is `tracemem(1:10)` not useful?

A: When `1:10` is called an object with an address in memory is created, but it is not bound to a name. Therefore, the object cannot be called or manipulated

from R. As no copies will be made, it is not useful to track the object for copying.

```
obj_addr(1:10)  # the object exists, but has no name
#> [1] "0x18f3d2f8"
```

Q2: Explain why `tracemem()` shows two copies when you run this code. Hint: carefully look at the difference between this code and the code shown earlier in the section.

```
x <- c(1L, 2L, 3L)
tracemem(x)

x[[3]] <- 4
```

A: Initially the vector `x` has integer type. The replacement call assigns a double to the third element of `x`, which triggers copy-on-modify.

```
x <- c(1L, 2L, 3L)
tracemem(x)
#> <0x66a4a70>

x[[3]] <- 4
#> tracemem[0x55eec7b3af38 -> 0x55eec774cc18]:
```

We can avoid the copy by sub-assigning an integer instead of a double:

```
x <- c(1L, 2L, 3L)
tracemem(x)
#> <0x55eec6940ae0>

x[[3]] <- 4L
```

Please be aware that running this code in RStudio will result in additional copies because of the reference from the environment pane.

Q3: Sketch out the relationship between the following objects: