Statistical Programming Methods with R

James Balamuta

2018-05-05

Contents

Part I

Preface

Welcome

Welcome to the Statistical Programming Methods with *R* textbook. Within this section, we'll go over the objective of the text, active development disclaimer, ways to contribute, conventions used, acknowledge those who have made contributions, and specify the license that the textbook is released under.

Objective of the Text

The book is meant to be used as primary reference material for developing statistical code within the *R* programming language. However, the goal is *not* to emphasize statistical theory as much as it is to explore underlying computational topics that power the algorithm. The thought behind this approach is to promote savviness in programming methods that are useful to approach a wide variety of analysis that Statisticians are expected to perform.

Presently, the book is used within the class STAT 385: Statistics Programming Methods offered at the University of Illinois at Urbana-Champaign (UIUC).

Active Development Disclaimer

The contents and structure of this book is under active development. Therefore, the best way to view the content will be to always access the book online rather than a printed copy to be sure you are using the latest version. The online version also affords additional features over the traditional PDF copy such as a scaling text, variety of font faces, and themed backgrounds. However, if you are in need of a local copy, a **pdf version** is also available.

Having said this, please note that the document is likely to contain many errors.

Your copy of this book was built 2018-05-05.

Contributing

If you notice any errors, we would be grateful if you would let us know. To let us know about the errors, there are two options available to you. The first and subsequently the fastest being if you are familiar with GitHub and know RMarkdown, then make a pull request and fix the issue yourself!. Within the online version, there is even an option to automatically start the pull request by clicking the edit button in the top-left corner of the text.

The second option, that will have a slightly slower resolution time is to send an email to `balamut2 AT illinois DOT edu` that includes: the error and a possible revision. Please put in the subject header: `[SPM]`.

Figure 1:

Rendering Mathematical Formulae

Throughout the book, there will be mathematical symbols used to express the material. Depending on the version of the book, there are two different render engines.

- For the online version, the text uses MathJax to render mathematical notation for the web. In the event the formulae does not load for a specific chapter, first try to refresh the page. 9 times out of 10 the issue is related to the software library not loading quickly.
- For the pdf version, the text is built using the recommended AMS LaTeX symbolic packages. As a result, there should be no issue displaying equations.

An example of a mathematical rendering capabilities would be given as:

$$a^2 + b^2 = c^2$$

R Code Conventions

The code used throughout the book will predominately be `R` code. To obtain a copy of `R`, go to the Comprehensive R Archive Network (CRAN) and download the appropriate installer for your operating system.

When `R` code is displayed it will be typeset using a `monospace` font with syntax highlighting enabled to ensure the differentiation of functions, variables, and so on. For example, the following adds 1 to 1

```
a = 1L + 1L
a
```

Each code segment may contain actual output from `R`. Such output will appear in grey font prefixed by `#>`. For example, the output of the above code segment would look like so:

```
#> [1] 2
```

Alongside the PDF download of the book, you should find the R code used within each chapter.

Acknowledgements

The text has been developed in the open and has benefited greatly from many people being able to alert the authors to problematic areas. We are greatful for the corrections, suggestions, or requests of clarity from the following:

- Your name and website address here!

License

Figure 2: This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

Chapter 1

The World of Programming

Chapter 2

R Programming Done Wrong

The motivation behind this chapter is to provide a reference guide to the more common mistakes when writing *R* code.

2.1 Beginners Luck

2.1.1 Checking for Equality vs. Assignment

The most common error by far that affects programmers is making an assignment when trying to check for equality (and vice versa)

```
# Assigning in `if`
if(x = 42) { cat("Life!") }
## Error: unexpected '=' in "if(x ="

# Correct
if(x == 42) { cat("Life!") }
```

```
# Equality Check instead of Assignment
x == 42
## No Error, but prints `TRUE` or `FALSE`

# Correct
x = 42
```

2.1.2 Missing object

```
x
Error: object 'x' not found

# Correct
x = 1
x
```

2.1.3 `if` vectorization usage

As emphasis on vectorization grows, there is a tendency to compare two vectors using the default `if()` instead of `ifelse()`

```
x = 1:5
y = 2:6
if(x > y) { TRUE } else { FALSE }
## Warning messages:
## In if (x > y) { : the condition has length > 1 and only the first element will be used

# Correct, if element-wise comparison required
ifelse(x > y, TRUE, FALSE)

# Correct, if totality-comparison requested
if(all(x > y)) { TRUE } else { FALSE }
```

2.1.4 Vector Recycling

Sometimes the length of vectors are not equal or the data does not divide evenly or oddly when perform a vectorized computation.

```
x = 1:5
y = 2:3

x + y
## Warning message:
## In x + y : longer object length is not a multiple
## of shorter object length

# Correct
x = 1:4
y = 2:3
x + y
# Repeats y twice
# 1 + 2, 2 + 3, 3 + 2, 4 + 3
```

2.2 Omitting Symbols

2.2.1 Mismatched curly brackets `{}`, parentheses `()`, or brackets `[]`

Often it is ideal to use parentheses or curly brackets for order of operations, though this sometimes causes a mismatch. A mismatch may also be present with the brackets subset operator `[]`

```
2*(x + y))
## Error: unexpected ')' in "2*(x + y))"

2*{x + y}}
## Error: unexpected '}' in "2*{x + y}}"

x]1
```

```
## Error: unexpected ']' in "x]"

# Corrected
2*((x + y))

2*{{x + y}}

x[1]
```

2.2.2 No Multiplication Symbol

When working on computations, sometimes we just "slip" and opt not to write a multiplication sign thinking the interpreter can understand the context.

```
2x+4
## Error: unexpected symbol in "2x"

# Correct
2*x + 4
```

2.2.3 Manual Data Entry

Sometimes it's easier as we'll see next week to manually enter data. The issue with this is sometimes you forget simple things like a `,`.

```
c(1, 2 3, 4)
## Error: unexpected numeric constant in "c(1,2 3"

# Correct
c(1, 2, 3, 4)
```

2.2.4 Strings in character values

At times, there may come a need to place a quotation inside of a string. To do this, requires using an escape character `\` or using `''` instead.

```
"toad"princess"
## Error: unexpected symbol in ""toad"princess"

# Corrected
"toad\"princess"
'toad"princess'
```

2.3 Special Values

2.3.1 Handling Missing Value Operations

The `NA` character indicates the presence of a missing value. These missing values can play havoc with computations.

```
x = c(1,NA,2)
3 + x
# No Error, but: [1]  4 NA  5

sum(x)
# No Error, but: [1] NA

# Corrected
1 + na.omit(x)    # Deletes NA
sum(x, na.rm = T) # Removes NA inside function
```

2.3.2 Finiteness of Values

R can have some funky finiteness problems due to how `NA` values are created.

```
x = c(NA,-Inf, Inf ,NaN)
is.na(x)
# No error, but: [1] TRUE FALSE FALSE TRUE

is.infinite(x)
# No error, but: [1] FALSE TRUE TRUE FALSE

# Correct
is.finite(x)
# [1] FALSE FALSE FALSE FALSE
```