Mastering R Markdown

Michael Harper

2018-08-07

Contents

Chapter 1

About the book

> This book is in a very early stage of development. If you have any suggestions on what should be included within this book, please get in touch via GitHub: https://github.com/mikey-harper/mastering-rmarkdown

- This book aims to bring together lots of useful tips for R Markdown.
- One of the common criticisms of markdown as a language is that it naturally limits what you can write. Users often therefore feel limited in what they can achieve with advanced customisation. However, we can easily edit.

1.1 Recommended Reading

There are already several fantastic books out there which you may have already read:

- Dynamic documents and knitr
- R Markdown: The Definitive guide
- Authoring books with bookdown
- Blogdown

Chapter 2

Introduction

- Motivation for book
- What readers can learn
- Feedback and errata
- Suggested examples

2.1 Overview of Book

- Explain the structure of the book

Chapter 3

Basics

- What should be known before reading this book
- Recommended reading
- `bookdown` is used as the default engine as it provides improvements

3.1 Quick Tips

- Short index of some basic ideas on how to improve R Markdown documents
- Include current date in R Markdown https://stackoverflow.com/questions/23449319/yaml-current-date-in-rmarkdown/23529410#23529410

Chapter 4

Managing Big Projects

- Practical tips on how a big project should be managed
- `source` is particularly useful for loading external scripts so that the R Markdown project isn't too bloated with code.

Ideas: - Use `(ref:tag)` to store page formatting options which might need to be reused. For example a page break

4.1 Sourcing Files

A benefit of using R Markdown is that it is easy

```
source("yourScript.R")
```

4.2 Caching

- Caching
- Ways it can be tailored to suit analysis. This cache invalidation is a great example: https://stackoverflow.com/questions/18376008/invalidate-a-chunks-cache-when-uncached-chunk-changes

4.3 Notifications

- Can link R Markdown with notifications if have long analysis

Chapter 5

LaTeX

For many authors, the main of long reports or books, the primary output will be LaTeX. In this chapter, we discuss approaches which can be used to customise the output of PDF reports.

Users should approach with a note of caution. One of the major benefits of R Markdown is

5.1 Inserting Commands

5.2 LaTeX preamble

5.3 Multi-figure plots

If we need to print multiple output graphs or figures, there are several ways this can be achieved.

5.4 Cross-output

As explained in https://bookdown.org/yihui/rmarkdown/r-code.html#figures of **?**, we can place multiple figures side-by-side using the `fig.hold='hold'` along with the `out.width` option. As an example below, we have set the `out.width="50%"`:

```
plot(1:10)
plot(rnorm(10), pch=19)
```

The main benefits of this approach is that it is easily achieved, and also works for both PDF and HTML outputs.

5.5 LaTeX subfigures

When writing a document you may want to include some slightly more complicated figures with multiple images. Subfigures are a useful LaTeX feature which allows us to achieve this by plotting multiple figures within a single plot and providing each with their own subcaption.

Subfigures require the LaTeX package `subfig`. The line `\usepackage{subfig}` must therefore be included within the YAML, or if you are using an external tex template you can add this to that file. For example:

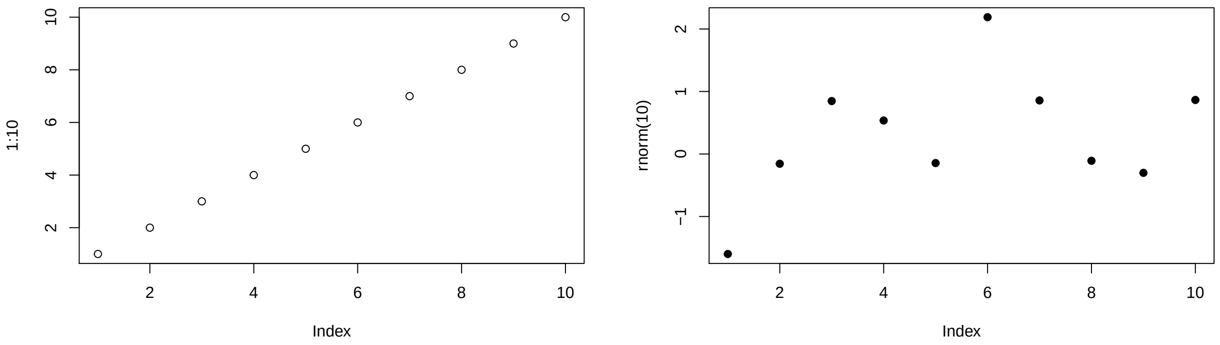


Figure 5.1: Side-by-side figures

As listed within the knitr chunk options, subfigures require a few additional settings to be set in the chunk header:

- `fig.subcap` is a list of the captions for subfigures
- `fig.ncol`: the number of columns of subfigures
- `out.width`: the output width of the figures. You will normally set this 100% divided by the number of sub columns.

An example is demonstrated below:

~~~~
```yaml
output: pdf_document
header-includes:
   - \usepackage{subfig}
```

```{r fig-sub, fig.cap='two plots', fig.subcap=c('one plot', 'the other one'), out.width='.49\\linewidt
plot(1:10)
plot(rnorm(10), pch=19)
```
~~~~

The output is shown in Figure 5.2.

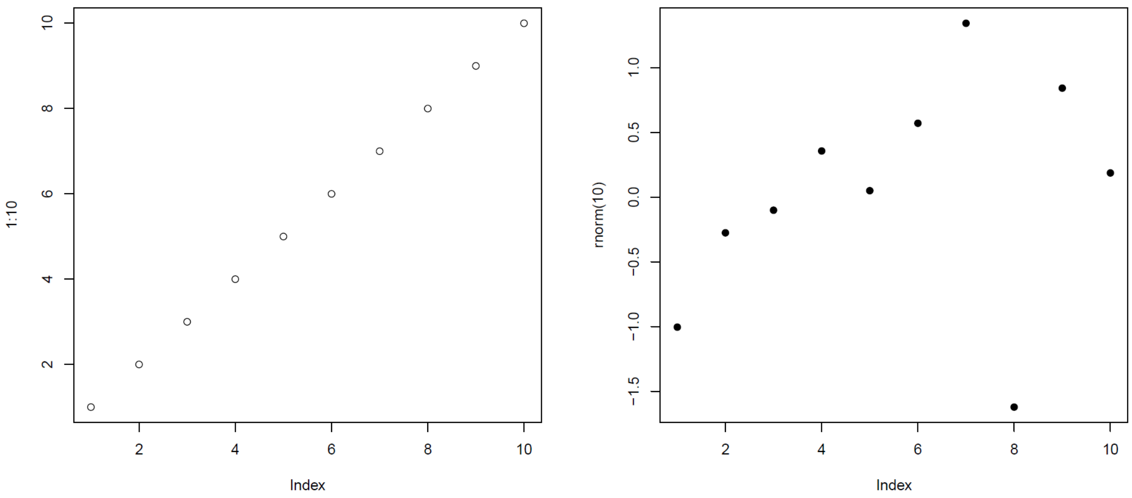
```r
knitr::include_graphics("images/subfigure.png", dpi = NA)
```

5.5.1 Using with list

5.6 Changing citation Engine

5.7 Altering Citation Style

- Using short author citations: https://stackoverflow.com/questions/48303890/using-short-author-citations-in-bookdown utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa



(a) one plot

(b) the other one

Figure 1: two plots

Figure 5.2: An example subcaption

Chapter 6

HTML Output

- Take some of the details from here: https://rmarkdown.rstudio.com/html_document_format.html

6.1 Tabbed headings

https://stackoverflow.com/questions/38062706/rmarkdown-tabbed-and-untabbed-headings?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa

Chapter 7

Multi-format projects

- One of the main benefits of R Markdown is that it is easy to create documents in a single source file and generate PDF, HTML, ebook and Word documents. This book example is available in all three formats. Project should therefore ideally be designed to be flexible for the multiple outputs. However, users will often find themselves wanting to fine-tune the output, and doing so will often require -Designing custom behavior which can respond to change in outputs using `is_latex_output` etc. For example, you may want to have interactive tables using `DT:datatable` in the HTML output but print static versions in the PDF.

7.1 Output-specific functions

- **bookdown** will take screenshots of HTML widgets in static reports

```r
library(ggplot2)
library(plotly)
p <- ggplot(data = diamonds, aes(x = cut, fill = clarity)) +
            geom_bar(position = "dodge")
ggplotly(p)
```

7.2 Designing output

If we wish to design our own format-specific function we can use the functions `knitr::is_latex_output()` `and`knitr::is_html_output‘. These function will return a TRUE/FALSE action

```r
knitr::is_html_output()
```

Chapter 8

Knitr Hooks

- Knitr hooks: these are explained within previous R Markdown book
- Include practical examples could be very useful for readers to see the power of it.
- Use some of these: https://gist.github.com/yihui/2629886

Chapter 9

Tables

- Using kableExtra to style tables
- Working with non R users
- Use HTML outputs and interactive graphics

Part I

Additional Packages

Chapter 10

Diagrams

It is often useful to express analysis using flowcharts and diagrams within reports. Although there are many separate programs which can be used to produce these, it can be beneficial to create these within our analysis directly. This makes it easier to update and edit the graph if we need to edit it in the future, and allows us to use the results from our analysis directly within the diagrams, making them more information.

While there are several different packages available for R, we will focus on the package **DiagrammeR** (Iannone, 2018). We also recommend that you read at http://rich-iannone.github.io/DiagrammeR/index.html . In this chapter we will explain some of the basic usages of the package.

10.1 Basic Diagrams

DiagrammeR provides methods to build graphs for a number of different graphing languages.

R Studio provides native support for Graphviz (`.gv`) and mermaid (`.mmd`) files. Using files of these types in RStudio provides the advantage of syntax coloring and allowing a quick preview of the diagram. For example, we can make a simple flowchart with the following code:

```
DiagrammeR::grViz("digraph {

        graph [layout = dot, rankdir = LR]

        node [shape = rectangle]
        rec1 [label = 'Step 1']
        rec2 [label = 'Step 2']
        rec3 [label =  'Step 3']
        rec4 [label = 'Step 4']

        # edge definitions with the node IDs
        rec1 -> rec2 -> rec3 -> rec4
        }",
        height = 200)
```

There are extensive controls which can be used to control the shape of nodes, colours, line types and add additional parameters.

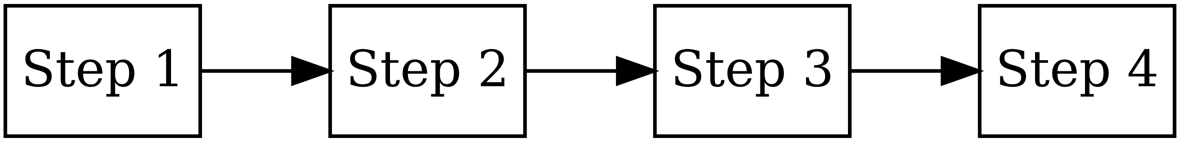


Figure 10.1: A basic graphic using DiagrammeR

10.2 Adding parameters to plots

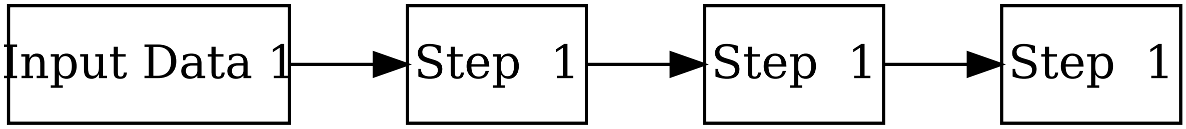
We can allow

Graphviz substitution allows for mixing in R expressions into a Graphviz graph specification without sacrificing readability.

If you specify a subsitution with `@@`, you must ensure there is a valid expression for that substitution. The expressions are placed as footnotes and their evaluations must result in an R vector object (i.e., not a data frame, list, or matrix). Because there is the possibility to have multiple substitutions, numbering is required. Thus, the @@ notation is immediately followed by a number and that number should correspond to the number of the footnoted R expression.

For example, suppose we have a dataset which is being analysed, and we would like to identify how many values are removed at each stage of a process.

```
DiagrammeR::grViz("
digraph graph2 {

graph [layout = dot, rankdir = LR]

# node definitions with substituted label text
node [shape = rectangle]
a [label = '@@1']
b [label = '@@2']
c [label = '@@3']
d [label = '@@4']

a -> b -> c -> d

}

[1]: paste('Input Data', 1)
[2]: paste('Step ', 1)
[3]: paste('Step ', 1)
[4]: paste('Step ', 1)
")
```


Input Data 1
Step 1
Step 1
Step 1

Chapter 11

References

Bibliography

Iannone, R. (2018). *DiagrammeR: Graph/Network Visualization*. R package version 1.0.0.