Introduction to R Markdown

Unit of Applied Statistics - SLU Alnarp

Contents

1 Introduction

These notes are intended to be a short introduction to R Markdown, which are a set of functions by which a user can incorporate free text and R code. The benefits of this approach is that it allows for quick changes of a manuscript and a safer transfer from analysis output to result presentation.

The Rstudio webpage for R Markdown (link) is the natural source for more information. An instruction for Markdown is available here.

2 Tools

The following is necessary or recommended in order to write R Markdown scripts and render to documents:

- the R package `knitr`,
- the R package `rmarkdown`,
- Pandoc,
- A Tex installation.

In short, `knitr` and `rmarkdown` transform a script with text and R code to a simpler text format called Markdown. This Markdown file is then rendered to a suitable output format by Pandoc. A Tex installation is necessary if this output format is a pdf file.

There a many possible ways to compile Tex. Miktex, (link), is a common recommendation for Windows users.

The R packages can be installed in the usual way (`install.packages()`).

Pandoc is a free document converter that can handle many different formats. Chances are it is already installed on your computer, as it is included in later installations of RStudio. It can otherwise be downloaded and installed from https://pandoc.org/.

3 The Script

3.1 Sections

An R Markdown script is a simple text file consisting of three parts: a header which specifies some general properties of the document; chunks which contain R code to be evaluated; and Markdown text containing the written text of the document.

3.2 Header

The header begins the document. It is contained in two lines made of three hyphens. A header may contain the title of the document, the authors name, and the type of document which should be rendered. The following example specifies that the document should be a pdf.

```
---
title: "The Mud King's Daughter"
author: Birgitta Trotzig
output: pdf_document
---
```

Exchanging `pdf_document` for `html_document`or `word_document` would render an html file and a word file respectively.

The header can be used to set a large number of options, some of which will appear later in these notes. The exact syntax of these options can be confusing and differ between document types. As an example, the following gives a pdf with a table of content:

```
---
title: "The Mud King's Daughter"
author: Birgitta Trotzig
output: pdf_document
toc: true
---
```

While the following gives an html page with a table of content:

```
---
title: "The Mud King's Daughter"
author: Birgitta Trotzig
output:
  html_document:
    toc: true
---
```

The line after output is indented with two spaces, and the line after that is indented a further two spaces.

The options syntax for a table of content in a Word document is similar to the html example.

3.3 Text

The free-flowing text is written in a mark-up language called Markdown. A mark-up language is a way to create a text document as a coded script. Unlike Word and other word processors, the parts of the document are specified in special syntax. The most well-known example of a mark-up language is likely to be html - the language of a standard website. In html, the parts are seperated using angle brackets, for example `<h1>Introduction</h1>` creates a header of level one which states "Introduction".

Useful syntax guides for Markdown are available here and here.

A brief example is given below. The headers are supressed from the printed results.

```
# A Level 1 Header
```

```
## A Level 2 Header
```

```
Text in *cursive*
```

Text in *cursive*

```
Text in **bold**
```

Text in **bold**

```
A web link in html documents is written
as [text with link](http://linkedsite.com/)
```

A web link in html documents is written as text with link

```
A list:

- First item.
- Second item.
```

A list:

- First item.
- Second item.

```
A table:

|Header 1   |Header 2   |
|-----------|-----------|
|Item 1     |Value 1    |
|Item 2     |Value 2    |
```

A table:

Header 1	Header 2
Item 1	Value 1
Item 2	Value 2

3.4 Chunks

R code is included using *chunks*. A chunk is initialized using three ticks and the option `{r}`. The chunk is closed with three ticks. An example is given below.

~~~
```{r}

R code goes here

```
~~~

In RStudio a chunk can be inserted into a script using ctrl + alt + I.

Specific options may be set in the curly bracket, for example `{r, echo = FALSE}` supresses the printout of the code so that only the results are shown.

R code can also be included in the text by writing a tick followed by the letter `r`. The line ends with the next tick mark:

```
`r R_code`
```

3.4.1 Tables

There are functions to produce tables in the knitr package (and probably several more). An example on the knitr::kable function:

```r
dat <- data.frame(Treatment = LETTERS[1:4],
                  Mean = round( runif(4), 3),
                  SE = round( runif(4) / 10, 3)
)

knitr::kable(dat,
             caption = "Mean value and
             standard error per treatment.")
```

Table 2: Mean value and standard error per treatment.

Treatment	Mean	SE
A	0.620	0.028
B	0.398	0.040
C	0.084	0.007
D	0.725	0.047

There is a small trick to make a internal reference to a table. It uses some Tex syntax as part of the caption:

```r
knitr::kable(dat,
             caption = "\\label{tab_means}Mean value and
             standard error per treatment.")
```

Table 3: Mean value and standard error per treatment.

Treatment	Mean	SE
A	0.620	0.028
B	0.398	0.040
C	0.084	0.007
D	0.725	0.047

We can then create a reference in-text using \ref{tab_means}. The result is a reference to Table 3. There is also a later solution in the bookmark package. Perhaps it is more stable.

It seems these solutions only work if the output file is a pdf. One possible solution for Word output is to use a vector of indices, say tableIndex. Everytime a new table is added to the document its label is added to the vector using tableIndex <-

`c(tableIndex, "tab_Means")`. Now the tables can be referenced using the label as

```
`r which(tableIndex == "tab_Means")`
```

This can also be used to specify the table number in the table heading by setting

```
caption = paste("Table ",
          which(tableIndex == "tab_Means"),
          ". Table header...", sep = "")
```

3.4.2 Figures

Figures created in R are written in a chunk and rendered in the document. The caption and label can be written in the chunk options. For example:

~~~
```{r, fig.cap = "\label{fig_sine}Sine w. trend.", fig.lp = "fig_sine"}

dat <- data.frame(x = sort(runif(500, -pi, pi)))

dat$y <- sin(dat$x) + 0.2 * dat$x + rnorm(500, sd = 0.1)

mod <- lm(y ~ 1 + x + I(x^2) + I(x^3) + I(x^4), dat)

plot(y ~ x, dat, pch = `.`)

lines(dat$x, predict(mod), lwd = 2)

```
~~~

Note the use of label to get a name for internal reference to the figure. As for the table, this is a bit of a trick and may not be completely fool-proof. The figure can be referenced using `\ref{fig_sine}`, resulting in a reference to Figure 1.

For Word documents, it is possible to number figures using the same method as for tables.

3.5 References

The handling of references depend on the way they are stored. One possibility is to keep ones references in a bib file - a simple text format used to handle references in Latex. The file is specified as an option in the header of the R Markdown file as `bibliography: name_of_file.bib`. This requires that the file is stored in the same folder as the R Markdown file. The references in the bib file can be accessed in the text using the syntax `@Schnabel1938` where `Schnabel1938` refers to a specific element stored in the file (in this case a classic article by Zoe Emily Schnabel). The result is the in-text reference Schnabel (1938) and a full reference at the end of the rendered text. References are put in brackets using `[@Schnabel1938]`, giving (Schnabel 1938). For a different citation, change the reference. For example, `@Legendre2012` and `@Yu015` produces references to Legendre (2012) and Yu et al. (2015). The full references appear in the reference section in the end.

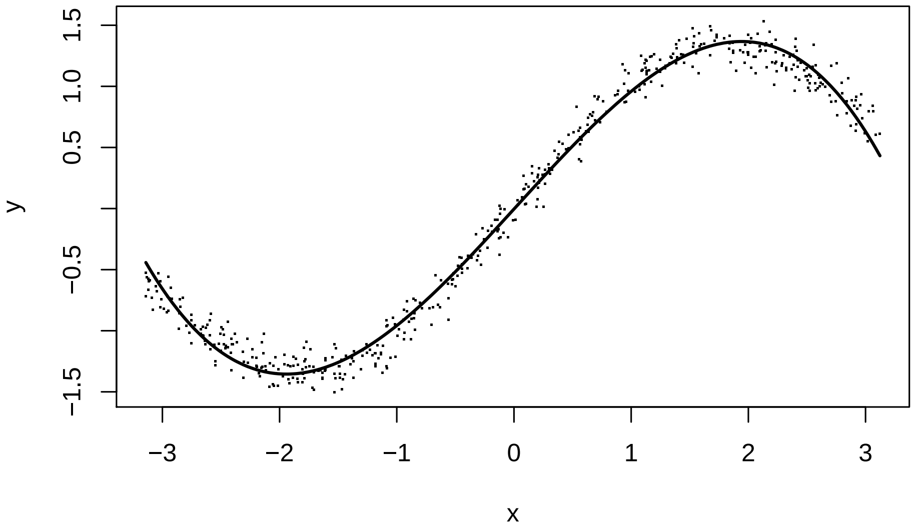

Figure 1: Sine function with linear trend.

For references in Endnote, it seems the only clear way is to export the references from Endnote to an .xml file,. By specifying the xml file as the bibliography using the header option `bibliography: name_of_file.xml`, referencing should work in the same way as the .bib case. Note that the reference labels can be set in Endnote under the label option.

3.6 Equations

In-line mathematical expression are written in Tex syntax and start and end with a dollar sign. For example `$y_i = \beta_0 + \beta_1 x_i + \varepsilon$` gives the standard one-variate regression function $y_i = \beta_0 + \beta_1 x_i + \varepsilon$.

Double dollar-signs give a free-standing equation, so `$${N \choose k} = \frac{N!}{(N-k)! k!}$$` gives

$${N \choose k} = \frac{N!}{(N-k)!k!}$$

Another alternative is to specify an equation environment using `\begin{equation}` and `\end{equation}`. Again, this is Tex syntax.

4 Rendering

There are two standard ways of rendering the file in the desired output format. For Rstudio users there is a knit button at the top of the script if the script is saved as an Rmd file. The arrow next to the button will open some options, in particular the choice of type of file to render (the usual choices are html, docx and pdf formats).

An alternative to this is to use the `render` function in the `rmarkdown` package.[1] The function is called in the standard way. The first option is the input file (i.e. the rmd file). If only the first input option is set, the output will be decided based on the information in the header and the output file will have the same name as the input file.

5 Problems

My most common mistake is to try to knit/render a file while the output file is open in another program (a pdf reader, the web browser or MS Word), but one can run into more complicated problems.

[1] The knit button is in fact a shortcut to the `render` function, but due to the settings, the results of using the button can differ from those of using the function.

It seems the SLU external storage solution can cause some problems, altough it depends on what file type one is rendering to. One solution is to install relevant packages (`rmarkdown`, `knitr`, and for presentations `revealjs`) on the local drive using the `lib` option in the `install.packages` function. When the packages are called using `library`, the same folder on the local drive should be used as the `lib.loc` option.

6 Templates

It is possible to set the layout of the rendered document using templates. For pdf files, the r package `rticles` comes with a number of preset templates. The template is specified in the header using `output: desired_template`, where the template must be available on the system. There is also an SLU thesis template, created by Rob Hart and available through the library. It is possible to use that template and Markdown, but I think it would be easier to use it directly in Latex. Similar to R Markdown, there is a possibility to incorporate R code in a Tex file. These are called Sweave files and the file extension is Rnw.

For Word files, a template is specified in the header by

```
---
title: "The Mud King's Daughter"
author: Birgitta Trotzig
output:
  word_document:
    reference_docx: refdoc.docx
---
```

Here, the style of refdoc.docx will be used for the rendered Word file. The refdoc.docx file should be located in the same folder as the R Markdown script.

References

Legendre, Pierre. 2012. *Numerical Ecology.* 3rd ed. Developments in Environmental Modelling, 24. Amsterdam: Elsevier.

Schnabel, Zoe Emily. 1938. "The Estimation of Total Fish Population of a Lake." *The American Mathematical Monthly* 45 (6). Mathematical Association of America:348–52.

Yu, Yang, Tjeerd-Jan Stomph, David Makowski, and Wopke van Der Werf. 2015. "Temporal Niche Differentiation Increases the Land Equivalent Ratio of Annual Intercrops: A Meta-Analysis." *Field Crops Research* 184. Elsevier:133–44.