R markdown — A powerful tool for report preparation

A. Jonathan R. Godfrey

11-15 July 2017

Abstract

In many research, educational and employment situations, we need to incorporate statistical analyses or graphs into a report of some kind. This workshop builds on the "Introduction to R" workshop to show you how you can do this for yourself. You'll see that you can update your report quickly and efficiently, no matter if the problem you find is small or large. After attending this workshop, you will be able to:

- include the results of calculations in your report,
- Include graphs of your data in your report,
- Include tabulated results or other text output from statistical analyses in your report.

The techniques shown in this workshop are especially valuable for reproducing reports that respond to updated data becoming available.

Getting started

We assume R is installed. It is a simple installation exercise. Get the latest installer for R under Windows. Users of other operating systems will need to find the appropriate version at CRAN - the Comprehensive R Archive Network. We also need a document conversion toolkit called pandoc. If you have not installed it already you can obtain a copy from the pandoc download page.

You should also have the add-on package known as BrailleR installed. Having BrailleR installed ensures that the other add-on packages we use in this session are also installed.

What is markdown?

Markdown is a very simple method for presenting formatted information from a source file that is just plain text. There are several flavours of markdown, but the common features are fairly simple to learn. A markdown document is converted into a variety of formats at the user's choosing; we will start by converting our markdown documents into HTML today because that is the most accessible format on offer.

Some formatting is simple to create, such as the use of number signs (sometimes called hash marks or pound) which generate headings. The level of the heading is determined by the number of hash symbols used. We will see only a few more formatting commands (actually not commands, more like typing practices) today.

The syntax is much simpler than LaTeX so markdown is a great way of getting simple jobs like assignments done with an acceptable level of presentation. For those jobs where a more professional look is required, such as a thesis or dissertation, an author would probably prefer to use LaTeX. All mathematical content is entered in markdown documents using standard LaTeX structures. If a document is started in markdown, it is easy to convert it to LaTeX when that is necessary.

What is R markdown?

R markdown is markdown with some added functionality that allows us to enter R code and have it processed so that the textual and graphical output we want is generated and pushed into the final document.

The exact process is as follows: The author creates the R markdown document in plain text (usually having an Rmd extension). The Rmd file is processed by R so that the code is evaluated and the right output is added to the document in the right places so that we now have a standard markdown file (usually having an md extension). The markdown file is then converted into the desired output file formats (commonly HTML, pdf via LaTeX, MS Word, or various slide presentation options).

A very convenient command will be used inside R to process any Rmd file. When you load the BrailleR package, you will hear that another package called knitr is being attached. The knitr package is the one that will process our files today, but other options for doing things in more fancy ways also exist.

The knitr package, the rmarkdown package and a suite of related packages form a key component of RStudio which is the most used integrated development environment (IDE) for R. Screen reader users cannot use RStudio at this time. The BrailleR package hopes to help blind users get access to the elements of RStudio that will make their lives easier.

A first glance at what is possible

In R, please make sure the BrailleR package is loaded using:

```
library(BrailleR)
```

and that you are aware of the current working directory, using:

```
getwd()
```

before you ask for the first example to be generated by the BrailleR package, using:

```
example(UniDesc)
```

This runs some code and creates a few graphs. You will be asked to hit the Enter key a few times to get to the next graph. Please do so; the graphs aren't so important to blind users.

Now, please look in your working directory to see what files were created by the commands just issued. You should find an Rmd file, an R script, and an HTML file. Let's open the HTML file first.

When considering the accessibility of the HTML file for screen reader users, take note that we have structured headings, some nicely formatted tables, and each graphic on the page has an alt tag telling us what the graphic is. In addition, the text output of the page includes use of the VI() functionality introduced in the Introduction to R workshop.

How did the HTML get created?

The example file we have just looked at is a univariate description of the variable Ozone from the airquality dataset. If we had wanted to issue the command for this variable without the example, we would have typed:

```
UniDesc(Ozone)
```

This command would have automatically opened the HTML file in a browser for us because this is the default option for the BrailleR package. There are other options for the BrailleR package that we might look at soon, but did you notice that the name of the author was at the top of the document? You can change the author of the work being done on your computer using:

```
SetAuthor("")
```

Please insert your name (between the quote marks) using the format you want your audience to see.

The UniDesc() command issued created the Rmd file that is in your working directory. Open it in a text editor of your choice. (I often just use Notepad in Windows.) This Rmd file gets converted in two ways: the HTML document we've seen already, and an R script file that just contains the R commands used in the Rmd file.

Let's look at that Rmd file more closely now.

Including R functions in an Rmd file

There are only two ways to have R commands in the Rmd file that lead to them being processed. The commands are either in-line or in code chunks (or blocks if you like). Code chunks are the more common type, but we will see how useful the in-line commands can be later.

Code chunks

A code chunk starts and then ends with three accent grave symbols, sometimes called backticks by the good people at RStudio. The opening three graves get some details put in curly braces , but the closing three graves are put on a line of their own. The opening and closing lines of a chunk therefore look like:

```
```{r ChunkName}
...
```
```

The details put in the curly braces are important. They tell the processor how to work with any commands put in place of the ... on the second line of this chunk. Our chunk will be processed as if all commands that are in the chunk are R commands.

We will usually include lots of chunks in a document; it is a good idea to label each chunk with a different name so that we can quickly find errors or move around our document. I try to use meaningful names for my chunks when I can.

The first chunk

Let's look at the first chunk in our markdown document now. It looks something like:

```
```{r setup, purl=FALSE, include=FALSE}
knitr::opts_chunk$set(dev=c("png", "pdf", "postscript", "svg"))
knitr::opts_chunk$set(comment="", echo=FALSE, fig.path="Ozone/Ozone-", fig.width=7)
```
```

The first line says to process the commands in R, then that the chunk is to be labelled "setup", and then includes options for this chunk. The commands included in this chunk are for setting chunk options that will be use for all remaining chunks in the document.

The first command tells the package to create graphs in four different formats. You need to include the png file format for the HTML pages to work well. The other options are principally for LaTeX users who want them. The SVG file format is experimental at this stage.

The second set of options does a wider range of things. The "comment" has a default of three hash or number signs, that I want removed so nothing is in the quote marks for this argument. The second option (echo=FALSE) stops the R commands being printed. I don't need them getting in the way here, and they can be collected together in the R script if they are wanted.

The third option (fig.path) tells R where all those figures are to be saved. The last option (fig.width) says to have all figures created with a width of 7 (inches). There is an option for figure height too, but this will vary depending on the figure type so it is set when the graphs are being created in other chunks.

These settings are not compulsory for all Rmd documents. They are the ones that work for the BrailleR package convenience functions like UniDesc().

Standard chunks

Have a look at the second chunk now please. It looks something like:

```
Ozone.count = length(Ozone)
Ozone.unique = length(unique(Ozone))
Ozone.Nobs = sum(!is.na(Ozone))
Ozone.Nmiss = sum(is.na(Ozone))
Ozone.mean = mean(Ozone, na.rm = TRUE)
Ozone.tmean5 = mean(Ozone, trim =0.025, na.rm = TRUE)
Ozone.tmean10 = mean(Ozone, trim =0.05, na.rm = TRUE)
Ozone.IQR = IQR(Ozone, na.rm = TRUE)
Ozone.sd = sd(Ozone, na.rm = TRUE)
Ozone.var = var(Ozone, na.rm = TRUE)
```

The chunk calculates a number of commonly used basic statistics which are each stored using a distinct name. At this stage, none of these basic summary measures is printed — that comes later.

Of note is that these commands are all evaluated even though the commands were not printed out in the HTML file because we said not to echo the commands — we didn't say that we wanted to ignore the commands used.

The remainder of the chunks in the example Rmd file do the wide variety of tasks required to get the HTML document we've been looking at. Some chunks create graphs and some create tables. A small number create the LaTeX versions of tables and perhaps they are the most confusing ones. Let's pass them over until much later on in your markdown journey.

In-line commands

One of the simplest commands in R is to print something out. You only need to type the name of that something to get it displayed. The paragraph in the HTML document that details the counts of data in our sample is generated by issuing the commands to print the counts in amongst the words. Take a look at the HTML document and then the Rmd file that created it.

Each in-line R command starts with a single backtick and an "r" followed by a single space; the R command is closed off with another single backtick.

The second line of the example Rmd file has an R command in it. This command grabs the option that has been set for the author. We could have set the line in the Rmd file to just print the option as it stood during creation, but the way it reads now, the printing of the author name is dynamic.

A note on efficiency

It might look like a lot of work to use markdown, and yes at first that is true. The benefits may not be clear in the short term.

The first example of when the use of markdown will save you time is when you quite obviously make a mistake in your working that you only find a long time afterwards. Perhaps you missed the fact that there were typos in the data; perhaps there were things you should have considered early on in the analysis that you now need to include.

More commonly though, the data set you are working with changes due to things beyond your control. With markdown, the time taken to re-run the analysis is next to nothing. Using R scripts is better than using any dialogue based statistics software in this situation, but avoiding the pain and suffering of endless copy and paste exercises is probably the greatest time saver for the markdown user.

An exercise

We are now going to tamper with the example markdown file. Make a copy and rename it to something easy to remember.

First, open your copy of the markdown file and delete everything below the paragraph giving the details of the counts of the example data.

Second, write a paragraph for yourself that includes the details of the mean and standard deviation of the example data. Save and close your file. You should now get ready to turn this document into a few file formats!

Converting markdown to HTML and other file types

We can turn your new Rmd file into HTML with a single command. Try:

```
knit2html("MyFile.Rmd")
```

where you made sure to use your file's actual name not MyFile.Rmd! In R you will see that the file has been processed or an error if you made a mistake.

Fix any mistakes now, perhaps by asking for help.

Take a look at your HTML file. Is it what you expected? Perhaps your mean and standard deviations are recorded to many more decimal places than you would like! You can change that Rmd file to use the round() command to ensure that the mean is rounded to the desired number of decimal places. You would put round(before the mean and a comma and the number of places after it; e.g. ,2) gives you 2dp.

Take a look more carefully in your working directory. You should find that a file with the extension md has been created. This file is just markdown after all the R commands were processed. All the working has been done to get that file.

We can convert the markdown file into a few other formats using a command that calls pandoc. Try:

```
pandoc("MyFile.md", format="docx")
```

to get a MS Word file. Other options for output file types exist.

Question time

We should have time to ask questions at the workshop, but there are other ways to have your questions answered. Take note that the Let's Use R Now document is updated as often as I get time and certainly gets additions when I discover something new and want to share it. The latest version is available at the LURN home page.

You should also seriously consider joining the Blind R Users Group. This is an email list I established in 2015. Send an email to the NFB mail server.

The subject line "subscribe" should already be filled in for you, as well as the word "end" in the body of the message. A confirmation message will come back to the address from which you send this request.

I try to keep details of the work I'm doing on the BrailleR package up to date on the BrailleR homepage.

Ultimately, you may just wish to send me a message.

If you choose to send me a message, please do remind me who you are and where we met. This might help me tailor the reply I send back to you.