Untitled

tcarroll

8 February 2015

Reproducible R

author: MRC Clinical Sciences Centre date:http://mrccsc.github.io/r_course/reproducibleR.html width: 1440 height: 1100 autosize: true font-import: font-family: 'Slabo 27px', serif; css:style.css

Reproducible Research

> "Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to humans what we want the computer to do." – Donald E. Knuth, Literate Programming, 1984

Reproducible Research in R

Sometime in the future, I, or my successor, will need to understand what analysis i did here.

Using RStudio to make reproducible documents is very easy, so why not?

A very quick reproducible document in R

- Find your R script of interest.
- Add the sessionInfo() function to the last line.
- Click the "Compile Notebook function" -> Select HTML document as output format.

Creating documents from R scripts

type:section

From Scripts to Notes

So we have just seen the speed at which you can produce a report document from an R script using Rstudio.

Rstudio makes things easy but for fine control we need to look at what is going on within Rstudio.

Rstudio makes use of **rmarkdown** and **knitr** packages.

Defaults

Several packages offer methods to create notes from R scripts.

One of the simplest way to create a note in R is to use the **render()** function in **rmarkdown** package.

```
library(rmarkdown)
render("scripts/script.r")
```

Output from render()

By default the render() function will have created a html file in the current working directory.

Have a look at the result script.html in the scripts directory.

Controlling the output type from render()

The render() function takes the argument **output_format**

```
render("scripts/script.r", output_format="html_document")
```

Setting output directory for rendered documents

The arguments **output_file** and **output_dir** can be used to control where output is rendered to.

Note that file extension must be supplied.

```
render("scripts/script.r", output_format="html_document", output_file="myRenderedDoc.html",output_dir="
```

Adding comments and text.

In R we can use # as comments in the code. This is the most basic type of documentation for your code.

```
# Generate some random numbers
myRandNumbers <- rnorm(100,10,2)
```

Adding comments and text.

If we want to include comments as text then we can use a new comment type #'

```
#' this would be placed as code
# Generate some random numbers
myRandNumbers <- rnorm(100,10,2)
```

YAML metadata.

If we wish to control author, title and date, we can insert metadata into the script as YAML.

```
#' ---
#' title: "CWB making notes example"
#' author: "Tom Carroll"
#' date: "Day 3 of CWB"
#' ---
#' this would be placed as text in html
# Generate some random numbers (This is a comment with code)
myRandNumbers <- rnorm(100,10,2)
```

We will come back to YAML later.

Controlling R code evaluation in notes.

We can control how the output from R looks in our rendered documents.

Options are passed to R code by adding a line preceeding R code with the special comment #+. We will look at some options later but a useful example is fig.height and fig.width to control figure height and width in the document.

```
#' Some comments for text.
#+ fig.width=3, fig.height=3
myRandNumbers <- rnorm(100,10,2)
hist(myRandNumbers)
```

Exercise

- Have at the example notebook script "scriptWithNotebookExamples.r" in scripts directory.
- Open scriptToConvertToNote.r in scripts directory and save as new name.
- Add notes to this script and compile with render() function or through RStudio.

Markdown

Under the hood, R is creating an intermediate document in **Markdown** format.

Markdown is a mark up language containing plain text and allowing for conversion to multiple rich text document types.

Common formats markdown renders to are - - html - pdf - Word doc

Markdown

Markdown is often used as an intermediate document in conversion from one type to another.

Github and Sourceforge make use of Markdown syntax in their Readme files and renders these in their webpages.

https://github.com/github/markup/blob/master/README.md

Markdown syntax.

Markdown uses simple syntax to control text output.

This allows for the inclusion of font styles, text structures, images and code chunks.

Lets look at some simple syntax for markdown to help us understand the R documents output from RStudio.

Markdown syntax- New line

Markdown is written as plain text and ignores new lines.

To include a new line in markdown, end the previous line with two spaces.

```
This is my first line.  # comment shows line end
This would be a new line.
This wouldn't be a new line.
```

To start a new paragraph, leave a line of space.

```
This is my first paragraph.

This is my second paragraph
```

Markdown syntax- Font emphasis

Emphasis can be added to text in markdown documents using either the **_** or ___*___

```
Italics = _Italics_ or *Italics*
Bold  =  __Bold__ or **Bold**
```

Markdown syntax- Including external images

Figures or external images can be used in Markdown documents.
Files may be local or accessible from http URL.

```
![alt text](imgs/Dist.jpg)
![alt text](http://mrccsc.github.io/r_course/imgs/Dist.jpg)
```

Markdown syntax- Creating section headers

Section headers can be added to Markdown documents.

Headers follow the same conventions as used in HTML markup and can implemented at multiple levels of size. Section headers in Markdown are created by using the # symbol

```
# Top level section
## Middle level section
### Bottom level section
```

Markdown syntax- Lists

Lists can be created in Markdown using the ___*___ symbol.
Nested lists be specified with + symbol.

```
* First item
* Second item
+ Second item A
+ Second item B
```

Markdown syntax- Order lists

Lists can also include ordered numbers.

```
1. First item
2. Second item
+ Second item A
+ Second item B
```

Markdown syntax- Code chunks

In Markdown, text may be highlighted as if code by placing the text between ‘”.

```
The code used to produce plot was
'''
hist(rnorm(100))
'''
```

Markdown syntax- Code chunks

In Markdown, text may be highlighted as if code by placing the text between ‘”.

```
The code used to produce plot was
'''
hist(rnorm(100))
'''
```

Markdown syntax- HTML links

HTML links can be included in Markdown documents either by simply including address in text or by using [] for the phrase to add link to, followed the link in ()

```
http://mrccsc.github.io
[Github site](http://mrccsc.github.io)
```

Markdown syntax- Page breaks.

Markdown allows for the specification of page breaks in your document.
To specify a page break use 3 or more asterisks or dashes.

```
Before the first page break
***
Before the second page break
---
```

rMarkdown.

rMarkdown is a script type used in R to allow for the generation of Markdown from R code. rMarkdown files will typically have the extension **.Rmd**

rMarkdown allows for the inclusion of Markdown syntax around **chunks** of R code.

The output from running the R code can be tightly controled using rMarkdown, allowing for very neat integration of results with code used to generate them

knitr

The **knitr** packages is the main route to create documents from **.Rmd** files.

knitr was created by Yihui Xie to wrap and clean up issues with other tools to make dynamic documents.

http://yihui.name/knitr/

rMarkdown. From Markdown to rMarkdown

The transition from Markdown to rMarkdown is very simple. All Markdown syntax may be included and code to be evaluated in R placed between a special code chunk.

The code chunck containing R code to execute is specified by the inclusion of **{r}** as below.

```
My Markdown **syntax** here
'''{r}
hist(rnorm(1000))
'''

rMarkdown. Controlling R code output - eval
=======
Options may be included in the R code chunks.

An important option is to choose whether code will be run or is meant for display only. This can be con
```

‘”{r,eval=F} hist(rnorm(1000)) ‘”

```
rMarkdown. Controlling R code output - Displaying code.
=======

It may be that you wish to report just the results and not include the code used to generate them. This
```

‘”{r,echo=F} hist(rnorm(1000)) ‘”

```
rMarkdown. Controlling R code output - message and warnings
=======
R can produce a lot of output not related to your results. To control whether messages and warnings are

Loading libraries in rMarkdown is often somewhere you would specify these as FALSE.
```

‘”{r,warning=F,message=F} library(ggplot2) ‘”

```
rMarkdown. Controlling figure output.
=======

Control over figure heights and widths can be implemented in rMarkdown using the **fig.width** and **fig
```

‘”{r,fig.width=5,fig.height=5} hist(rnorm(100)) ‘”

```
rMarkdown. Automatically tidying code.
=======

The code within the **{r}** code block can be reformatted using the formatR package. This can be automat
```

‘”{r,tidy=T} hist(rnorm(100)) ‘”

```
rMarkdown. Placing code and output together
=======

The code within the **{r}** code block will by default appear in a separate block to results output. To
```

‘”{r,collapse=T} temp <- rnorm(10) temp ‘”

```
rMarkdown. Inserting tables.
=======

The results of printing data frames or matrices in the console aren't neat.

We can insert HTML tables into Markdown by setting the **results** option to **asis** and using the kni
```

‘”{r,results=‘asis’} temp <- rnorm(10) temp2 <- rnorm(10) dfExample <- cbind(temp,temp2) kable(dfExample) ‘”

```
rMarkdown. Evaluating code within markdown text.
=======

It may be useful to report the results of R within the block of Markdown. This can be done adding the co
```

Here is some freeform *markdown* and the first result from an rnorm call is ‘r rnorm(3)[1]’, followed by some more free form text.

```
rMarkdown: cache
===

Some operations may take a significant time or resource to compute.

The **cache** argument may be used to save the results in the current working directory. This code chun
```

‘”{r,cache=TRUE} x <- sample(1000,10ˆ8,replace=T) length(x) ‘”

```
YAML in rMarkdown.
=====

In rMarkdown the options for document processing are stored in YAML format at the top of the document.


Controlling output type.
=====

The **output** YAML option specifies the document type to be produced.
```

```
Figure options in YAML
===

Global default options for figure sizes and devices used can be set within the YAML metadata.
```

```
Adding styles
===
Styles for HTML can be applied using the **theme** option and syntax highlighting styles control by the


For a full list of theme options see -
http://rmarkdown.rstudio.com/html_document_format.html

Additional styles
=====
```

“‘

Custom styles can also be applied to rMarkdown documents using CSS style files and the **css** option.

Using Rstudio

Lets see how to do this in RStudio.

File -> New File -> R Markdown

Resources

http://yihui.name/knitr/
http://rmarkdown.rstudio.com/
http://rcharts.io/
http://rstudio.github.io/packrat/

Exercises

- Open up markdownExampleDefaultStyles.Rmd and markdownExample.Rmd in the scripts directory. Have a look at the rMarkdown examples here and the resulting output html files.

Example HTML Default style.
Example HTML with extra style.

- Open scriptToConvertToRMarkdown.r in scripts directory and save as new name.
- Convert this script to an Rmarkdown document using the render() function or inside RStudio.