R Markdown Sample

Your Name

Oct 8, 2021

Contents

The original document of this file is provided by Jeromy Anglim and modified by Kenya Amano and Inhwan Ko. Please visit the the link if you want to see the full version of this file.

```
# To briefly explain each element in the heading:
# title: "R Markdown Sample" # write down your title
# subtitle: "if you want to add a subtitle, please write here"
# author: "Your Name"
# date: "Oct 8, 2021"
# indent: T # turn off into "F" if you don't want an indent
# output: # dictates your output format
#   pdf_document: # when you want a pdf format output
#     latex_engine: xelatex # latex engine: use xelatex for tinytex
#     toc: T # table of contents
#   html_document: # when you want an html format output
#     df_print: paged # page numbers on an html format
# editor_options:
#   chunk_output_type: console  # outputs are shown in a console, not inline (here)
```

0. Intro

This post examines the features of R Markdown using knitr in Rstudio 1.3. This combination of tools provides an exciting improvement in usability for reproducible analysis. Specifically, this post:

1. discusses getting started with R Markdown and 'knitr' in Rstudio 1.3;
2. provides a basic example of producing console output and plots using R Markdown;
3. highlights several code chunk options such as caching and controlling how input and output is displayed;
4. demonstrates use of standard Markdown notation as well as the extended features of formulas and tables; and
5. discusses the implications of R Markdown. This post was produced with R Markdown. The source code is available here as a gist.

The post may be most useful if the source code and displayed post are viewed side by side. In some instances, I include a copy of the R Markdown in the displayed HTML, but most of the time I assume you are reading the source and post side by side.

1. Getting started

To work with R Markdown, if necessary:

- Install R
- Install the latest version of RStudio (at time of posting, this is 0.96)
- Install the latest version of the `knitr` package: `install.packages("knitr")`

To run the basic working example that produced this blog post:

- Open R Studio, and go to File - New - R Markdown
- If necessary install packages: Do 'install.packages("PackageName")
- Paste in the contents of this gist (which contains the R Markdown file used to produce this post) and save the file with an `.rmd` extension
- Click Knit HTML

To produce PDF file, you need TeX files. * Easy way: Install the `tinytex` package: `install.packages("tinytex")`. Then run `tinytex::install_tinytex()`. * If you want full version of TeX: For Mac install MacTeX. For Windows install TeX Live.

```
library(knitr)
library(tinytex)
```

- More info: R Markdown Reference Guide R Markdown Cheat Sheet

2. Prepare for analyses

```
set.seed(1234)

#install.packages("lattice")
#install.packages("stargazer")
#install.packages("pander")
```

```
library(tidyverse)
library(lattice)
library(stargazer)
library(pander)
```

Without specifying the options of chunk, you get *warning*

3. Basic console output

To insert an R code chunk, you can type it manually or just press `Chunks - Insert chunks` or use the shortcut key. This will produce the following code chunk:

Pressing tab when inside the braces will bring up code chunk options.

The following R code chunk labelled `basicconsole` is as follows:

````
```r
x <- 1:10
y <- round(rnorm(10, x, 1), 2)
df <- data.frame(x, y)
df
```

```
##     x     y
## 1   1 -0.21
## 2   2  2.28
## 3   3  4.08
## 4   4  1.65
## 5   5  5.43
## 6   6  6.51
## 7   7  6.43
## 8   8  7.45
## 9   9  8.44
## 10 10  9.11
```
````

The code chunk input and output is then displayed as follows:

```
x <- 1:10
y <- round(rnorm(10, x, 1), 2)
df <- data.frame(x, y)
df
```

```
##     x     y
## 1   1  0.52
## 2   2  1.00
## 3   3  2.22
## 4   4  4.06
## 5   5  5.96
## 6   6  5.89
## 7   7  6.49
## 8   8  7.09
## 9   9  8.16
## 10 10 12.42
```

```r
x <- 1:10
y <- round(rnorm(10, x, 1), 2)
df <- data.frame(x, y)
df
```

```
##     x    y
## 1   1 1.13
## 2   2 1.51
## 3   3 2.56
## 4   4 4.46
## 5   5 4.31
## 6   6 4.55
## 7   7 7.57
## 8   8 6.98
## 9   9 8.98
## 10 10 9.06
```

```r
x <- 1:10
y <- round(rnorm(10, x, 1), 2)
df <- data.frame(x, y)
df
```

```
##     x    y
## 1   1 2.10
## 2   2 1.52
## 3   3 2.29
## 4   4 3.50
## 5   5 3.37
## 6   6 4.83
## 7   7 4.82
## 8   8 6.66
## 9   9 8.71
## 10 10 9.53
```

Inline value can be shown as follows: The average of y is 4.733.

4. R Code chunk features

Create Markdown code from R

Frequently used chunk options

Option	Description
include	If FALSE, knitr will run the chunk but not include the chunk in the final document
echo	If FALSE, knitr will not display the code in the code chunk above it's results in the final document.
error	If FALSE, knitr will not display any error messages generated by the code.
message	If FALSE, knitr will not display any messages generated by the code.
warning	If FALSE, knitr will not display any warning messages generated by the code.

Recommendation for Homework:

Option	HW setting
include	TRUE
echo	TRUE
error	FALSE
message	FALSE
warning	FALSE

4.1. Echo and Results

The following code hides the command input (i.e., `echo=FALSE`), and outputs the content directly as code (i.e., `results=asis`).

Here are some dot points

- The value of y[1] is 2.1
- The value of y[2] is 1.52
- The value of y[3] is 2.29

This code includes the command input (i.e., `echo=TRUE`) with markup output (i.e., results -> default)

```
cat(paste("* The value of y[", 1:3, "] is ", y[1:3], sep="", collapse="\n"))
```

```
## * The value of y[1] is 2.1
## * The value of y[2] is 1.52
## * The value of y[3] is 2.29
```

4.2. Message and Warning

While the chunk without specification of options show all wanings and messages....

```
df %>%
  summarize_at(vars(y), funs(mean))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
##       y
## 1 4.733
```

this code does not output warnings

```
df %>%
  summarize_at(vars(y), funs(mean))
```

```
##       y
## 1 4.733
```

So, I recommend having this chunk at the very beginning of your RMarkdown when submitting an assignment:

knitr::opts_chunk$set(include=T, echo = T, error=F, message=F, warning=F)

and modify manually whenever necessary (e.g., add results='asis' when inserting a graph).

5. Cache analysis

Caching analyses is straightforward. Here's example code. On the first run on my computer, this took about 10 seconds. On subsequent runs, this code was not run.

If you want to rerun cached code chunks, just delete the contents of the `cache` folder

```
```r
for (i in 1:5000) {
    lm((i+1)~i)
}
```
```

6. Basic markdown functionality

For those not familiar with standard Markdown, the following may be useful. See the source code for how to produce such points. However, RStudio does include a Markdown quick reference button that adequatly covers this material.

Formatting a paragraph

If you simply press enter after this, this won't be moved to a next line.

If you press 2 spacebars after this,
this will be moved to a next line.

If you want to make a new paragraph after this (with some space by default),

Write a LaTeX function called `\par` in front of the new paragraph you want to start.

If you do not want an indent after the first paragraph of each section,
please set `F` for `indent` in the header.

Dot Points

Simple dot points:

- Point 1
- Point 2
- Point 3

and numeric dot points:

1. Number 1
2. Number 2
3. Number 3

and nested dot points (2 tabs):

- A

  - A.1
  - A.2
- B
  - B.1
  - B.2

Or use LaTex formats such as:

This is how you make a numbered list:

1. First element of list
2. Second element of list
3. Third element of list

This is how you make an unnumbered list:

- First element of list
- Second element of list

7. Equations

Equations are included by using LaTeX notation and including them either between single dollar signs (inline equations) or double dollar signs (displayed equations). If you hang around the Q&A site CrossValidated you'll be familiar with this idea.

There are inline equations such as $y_i = \alpha + \beta x_i + e_i$.

And displayed formulas:

$$\frac{1}{1 + \exp(-x)}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\begin{aligned} X &= (x + a)(x - b) \\ &= x(x - b) + a(x - b) \\ &= x^2 + x(a - b) - ab \end{aligned}$$

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix}$$

More info: LaTeX wiki

Chris suggests not to use dollar signs when writing up mathematical notations. This is because when producing an output, TeX often awkwardly interprets dollar signs and returns an error. However, if you are not sure about whether you used the right LaTeX codes, you can wrap notations around with dollar signs temporarily to see their outputs. Then erase them when *knitting* the R Markdown file. You can use `\begin{equation} ~ \end{equation}` or `\begin{eqnarray} ~ \end{eqnarray}` instead to wrap around your LaTeX mathematical notations.

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{pmatrix} \tag{1}$$

8. Tables

Tables can be included using the following notation

A	B	C
1	Male	Blue
2	Female	Pink

Or you want to show nice regression tables

```
Mod1 <- y ~ x
Res1 <-
  lm(formula = Mod1,
     data = df)

Mod2 <- y ~ x^2
Res2 <-
  lm(formula = Mod2,
     data = df)
```

```
stargazer(Res1, Res2)
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu % Date and time: , 10 08, 2021 - PM 3:48:48

Table 4:

	Dependent variable:	
	y	
	(1)	(2)
x	0.876***	0.876***
	(0.095)	(0.095)
Constant	−0.083	−0.083
	(0.587)	(0.587)
Observations	10	10
R^2	0.915	0.915
Adjusted R^2	0.904	0.904
Residual Std. Error (df = 8)	0.860	0.860
F Statistic (df = 1; 8)	85.568***	85.568***
Note:	*p<0.1; **p<0.05; ***p<0.01	

```
#For html
#stargazer(Res1, Res2, type = "html")
```

More info: Cheat Sheet
If you want to create a fancy table from data.frame, you can use “pander”

```
Table <-
df %>%
  mutate(z = if_else(y>5, 1, 0)) %>%
  t()

Table
```

```
##   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## x  1.0 2.00 3.00  4.0 5.00 6.00 7.00 8.00 9.00 10.00
## y  2.1 1.52 2.29  3.5 3.37 4.83 4.82 6.66 8.71  9.53
## z  0.0 0.00 0.00  0.0 0.00 0.00 0.00 1.00 1.00  1.00
```

With `pander`

```
Table  %>%
  pander(caption ="Fancy Table")
```

Table 5: Fancy Table

x	1	2	3	4	5	6	7	8	9	10
y	2.1	1.52	2.29	3.5	3.37	4.83	4.82	6.66	8.71	9.53
z	0	0	0	0	0	0	0	1	1	1