Markdown Magic

Juli Tkotz

05 Mai 2019

Some rmarkdown basics

Chunk options

We include code chunks in our document. We can specify the default chunk options using the `knitr` package which spares you some annoying copy-pasting. You can override them for individual chunks, though! Here, I want the raw code to be shown, so I set `echo = TRUE`. I also want the code to be executed, so I set `include = TRUE`. I also like to live dangerously and want to supress any warning messages. And I'd like my figures to be centered.

```
knitr::opts_chunk$set(echo = TRUE, include = TRUE, message = FALSE,
                      warning = FALSE, fig.align = "center")
```

Example: `echo = TRUE` and `include = TRUE`:

```
print("I like lasagna.")
```

```
## [1] "I like lasagna."
```

Compared to: `echo = FALSE` and `include = TRUE`:

```
## [1] "I like lasagna."
```

Compared to: `echo = TRUE` and `include = FALSE`:

Inline code

Let's get some data first. `R` and some of its packages come with some built-in data sets. We'll use the diamonds data set.

```
library(tidyverse)
diamonds <- diamonds
```

```
# We only have a "normal" PDF document here, but check the papaja package for
# an APA template and e.g. tables in APA format
# Don't forget to set results = 'asis'
# We also need the package booktabs in the header
library(xtable)

diamonds_table <- head(diamonds)
```

```
print(xtable(diamonds_table, comment = FALSE, include.rownames = FALSE,
             booktabs = TRUE))
```

% latex table generated in R 3.5.3 by xtable 1.8-3 package % Tue May 07 14:42:56 2019

Now, let's have some truly sparkling (get it?) markdown magic. You can throw code into your text that is directly executed. Use backticks and specify `R` as language (see .Rmd-file for how the code looks like). So, for example, we can include this code ...

```
round(mean(diamonds$price), 2)
```

```
## [1] 3932.8
```

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.50	55.00	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.80	61.00	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.90	65.00	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.40	58.00	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.30	58.00	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.80	57.00	336	3.94	3.96	2.48

... directly in our text, which looks like this: 3932.8. That comes in really handy when reporting statistics because you can just change the data source and all means, standard deviations and stuff you have in your text will change with it.

Printing statistical results

There are also packages which make printing of results from statistical tests easier. For example, if we run a t-test (and calculate the effect size) on the price difference between fair (lowest level) and ideal (highest level) diamonds, we can print the output readily (?) formatted (?) in APA style.

```
t_cut_price <- t.test(diamonds$price[diamonds$cut == "Fair"],
                      diamonds$price[diamonds$cut == "Ideal"])

library(effsize)

d_cut_price <- cohen.d(diamonds$price[diamonds$cut == "Fair"],
                       diamonds$price[diamonds$cut == "Ideal"])
```

```
# devtools, git and stuff
library(prmisc)

# Packages that are not released on CRAN, but in GitHub repositories, can be
# downloaded with the aid of the package devtools. In this case, the code would
# look like this:
#
# library("devtools")
# install_github("m-Py/prmisc")
```

It appears that the ideal cut diamonds ($M = 3457.54$, $SD = 3808.40$) are pricier than the fair cut ones ($M = 4358.76$, $SD = 3560.39$), $t(1894.79) = 9.75$, $p < .001$, $d = 0.24$.

Citations

There is a vast amount of packages offering functions like this. I just shamelessly promote this one because it is from Papenberg (2018) and I contributed a bit. Also, it is a nice example to show you how citations work. You can see that we included a nice little .bib-file in our header. Now, we can just call citations by calling @[1], following the name I gave the reference: `@R-prmisc`

Without brackets, it looks like what you saw above. Some brackets `[]` make it look like this (Papenberg 2018). It's always nice to cite the software you used! If you want to cite a package, you can get its citation in bibtex format by calling, `toBibtex(citation("prmisc"))` (with the name of the package that you want, obviously). Don't forget to give the reference a name when you save it in your .bib-file. Use ; within brackets to cite multiple sources (Kable and Glimcher 2007; Green et al. 1981). In a normal PDF document, references

[1] Latex is extremly picky about inserting a simple at symbol in text. Usually, escaping special characters with \ does the trick (in fact, I had to escape the \ itself), but not in this case. Wrapping it in dollar signs (= "math mode") works, but it's a dirty hack.

in text don't always follow APA guidelines, but when you use a `papaja` template, the package will work stuff out for you.

References

Green, Leonard, Edwin B. Fisher, Steven Perlow, and Lisa Sherman. 1981. "Preference Reversal and Self Control: Choice as a Function of Reward Amount and Delay." *Behavioural Processes* 1 (1): 43–51.

Kable, Joseph W., and Paul W. Glimcher. 2007. "The neural correlates of subjective value during intertemporal choice." *Nat. Neurosci.* 10 (12): 1625–33.

Papenberg, Martin. 2018. *Prmisc: Printing Statistical Results in R Markdown.* https://github.com/m-Py/prmisc.