R Markdown Lesson 01: Getting started

Alexander Strobel

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. In the following, I will briefly give a general outline of R Markdown, which is an easy to learn language for creating updatable and reproducible reports. In RStudio (RStudio Team, 2016), you first create a new file of format R Markdown, define some document-specific issues such as the title, author or the output format in the so-called YAML header, and then simply start writing! Formatting is quite straightforward. Some examples for formatting text are:

> # Heading 1
>
> ### Heading 2
>
> *italic text*
>
> **bold text**
>
> superscript/subscript: m^2/CO_2
>
> endash: –
>
> emdash: —
>
> equation: $A = \pi * r^2$
>
> reporting a Spearman correlations: $r_s = .25, p = .012$
>
> reporting results of a t-test : $t_{196} = -0.43, p = .668, d = -.06$
>
> reporting a power calculation: "Given $\alpha = .05$ (two-sided) and a power of $1 - \beta = .95$, $N = 138$ participants are needed to detect a $>$ correlation of $r = .30$."

You may have noticed two things in the code that generated the above examples. First, all lines start with an ">". I did this because I wanted the examples to be indented. Second, for the last three lines, I embedded the results in a double $ which results in the text being interpreted as LaTeX text, hence, it will be formatted that way. When you click the *Knit* button, a document will be generated that includes both the written content and the output of any embedded so-called R code chunks within the document. You can embed an R code chunk like this:

```
set.seed(242)       # set custom seed for reproducible results
x = rnorm(100)        # generate random variable x with 100 obs.
y = 0.2 * x + rnorm(100)  # generate random variable y with 100 obs. that covaries with x
cor(x, y)             # get Pearson correlation
```

```
## [1] 0.4370231
```

In this example, to have the same set of random variables and the same correlation returned each time, I set a custom seed, which is particularly important for arriving at reproducible results when doing simulations or bootstrapping. Here, I set the seed to 242, my favorite number (after all powers of two), but the number actually does not matter as long as it is always the same for the same analysis[1]. Unless for demonstration purposes, you will usually not want to have the output of the code chunk shown as above just as it would appear on the console. In this case, set echo = F at the beginning of the code chunk. Still, you will have access to the output of the code chunk, as shown here for a correlation analysis.

In this example, a code chunk is executed (which is not shown here), a correlation and the associated p-value are determined, and the correlation is classified as *small*, *medium* or *large*. Then, in the text, the result is reported by referencing the variables generated by the code chunk: "The variables were correlated with a large effect size, $r = .44$, $p < .001$". The sentence elements *large*, *.44* and *< .001* are placeholders filled with content by the output of the code chunk. The great thing here is that if you might want or being urged (by a supervisor or a reviewer) to include more detail on the statistics, you can do so by easily adding another bit of code ... (invisible code chunck executed here)

... and then write "The variables were correlated with a large effect size, $r = .44$, $t(98) = 4.81$, $p < .001$".

In the above code chunks, the classification of the effect size was done using a conditional *if* command. If you have multiple correlations to be classified, this would result in rather long code chunks. In case you repeatedly use the same code to perform some task, just write a function. Doing so even allows you to easily switch between different classifcation schemes (in this example via the arguments *method='gignac'*, or *method='cohen'*, respectively).

Here's the result: "The variables showed a correlation of $r = .44$, $p < .001$, i.e., they correlated with a large effect size according the classification scheme by Gignac and Szorodai (2016), and correlated with a medium effect size according the classification scheme by Cohen (1988).

Including Plots

You can also embed plots and adjust their appearance *ad libitum*, for example:

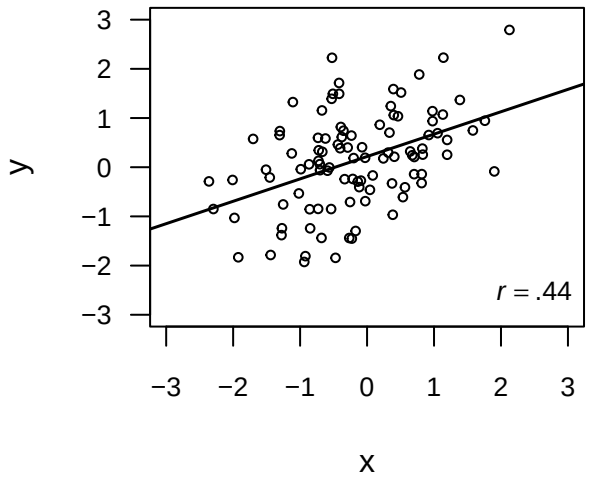

Figure 1. Scatterplot of the variables of interest.

[1]To arrive at comparable results when doing random number generation using a different software, you also need to make sure that the same kind of random number generator is used. Preferably, you make sure to use the *Mersenne-Twister* random number generator, it is the default in both R and SPSS. You can retrieve the random number generator currently used via *RNGkind()* and can set or reset it using this function as well, e.g., via *RNGkind('Mersenne-Twister, 'Inversion', 'Rounding)*. By the way: You now also know (a) how to insert footnotes – via "^[footnote text]", and (b) how certain prespecified commands like the one to insert footnotes are not executed, but printed as is – via a backslash ahead of the command, which in LaTeX/R Markdown is a so-called escape character. More on this matter in later lessons.

Interim Summary

So, by now, you should have learned how to

- set up an R Markdown file
- do simple text formatting such as headings of different depth, usage of italics and bold font, and writing simple formulae
- providing basic statistical results in an APA-compliant manner
- obtain results of statistical analyses within R code chunks and reference to these results in your text
- even pre-format parts of your results section based on your statistical results such as effect size classifications
- write functions that make reporting of results more convenient (and code shorter)
- include figures of your results in your document

Exercises

To exercise what you have learnt in this lesson, do the following:

1. Set up an R Markdown file with Word output and insert a subtitle in the YAML header.
2. Try to write 'partial eta-squared' using LaTeX code.
3. Use the variables x and y generated above to run a simple linear regression in a new code chunk and report the results in the body of the document including multiple R-squared.
4. Write a function that reports the means and standard deviations of x and y, either without or with a statement on whether the variables were normally distributed (as determined via a Shapiro-Wilk test, with $p > .20$ indicating no deviation from the normal distribution).
5. Insert a figure of a scatter plot of the regression of y on x including confidence intervals of the regression slope using the *predict()* function. Also set the figure caption in the code chunk alongside fig.with and fig.height using the argument fig.cap.

Further reading

For solving these tasks as well as all other questions around R Markdown, you can resort to the following resources (just click on it, these are links):

- Using R Markdown
- R Markdown Cheat Sheet
- R Markdown Cookbook

The latter is a very comprehensive guide on how to use R Markdown.

Outlook

In the next lesson, we will elaborate on the present lesson, especially on

- the LaTeX language – that we will sometime or often use alongside R Markdown – in more general
- setting up documents in a custom fashion
- formatting text in an advanced way and writing more complicated formulae
- providing a variety of statistical results in an advanced manner

References

Cohen, J., (1988). *Statistical power analysis for the behavioral sciences (2nd ed.)*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.

Gignac, G. E. & Szodorai, E. T. (2016). Effect size guidelines for individual differences researchers. *Personality and Individual Differences, 102*, 74-78.

RStudio Team (2016). *RStudio: Integrated development environment for R*. Boston, MA: RStudio, Inc. Retrieved from: http://www.rstudio.com/

Note: In this introductory lesson, references were inserted by hand. In the upcoming lessons, you will learn how to use a BibTeX library to handle citations.