Communicating Science using Rmarkown

Data Science Lecture Series: Advanced R

W. Evan Johnson, Ph.D.
Professor, Division of Infectious Disease
Director, Center for Data Science
Rutgers University – New Jersey Medical School

2023-03-27

Section 1

Communicating Science

Communicating Science

Effective science communicators educate non-specialist audiences about scientific topics, issues, and debates in ways that are informative, accessible, and empowering.

Science communicators should be able to answer the following questions:

- Who is my audience?
- What is my message for my audience?
- What medium am I going to use to communicate my message to my audience?

Communicating Science

The **lay public** is made up of all the people who are not experts in a specific field. When addressing the lay public:

- Keep the story simple and front-load exciting aspects
- Make the story relevant to your audience
- Use analogies and visuals
- Front-load the story
- Avoid jargon - simple language but don't oversimplify
- Include the people and the process (challenges, successes, collaborations, etc.)

Communicating Science

The **media** is a "mediator" between scientists and the public. Note the media is not a homogenous group.

Describing your process, challenges, successes, and collaborations are important for writing an informative and engaging press release. Read a few popular science articles to get a sense of how your research might eventually appear in the news and magazines.

Articles about your work should include visuals–videos or photos–that will draw readers' attention to the article and help them grasp the gist of the piece.

Communicating Science

Scientists can share their knowledge with **policy makers** through meetings, testimonies, and open presentations. Suggestions for communicating with policy makers:

- Know what issues policy makers are currently discussing and debating
- Keep your explanations simple and relevant
- Think of some actionable solutions to the problem
- Think about the problem and solution in the context of the policy maker's constituency
- Be confident in yourself and what you know
- Approach a meeting as a conversation, not a presentation
- Create a one-pager with your message and key points

Communicating Science

Visuals make the data supporting your message clear and accessible to your audience. Science visualizations include:

1. Graphs, tables, and infographics
2. Conceptual diagrams
3. Maps, Satellite photos
4. Photographs

Keep the following points in mind:

1. Use a consistent style and format
2. Use colors with purpose
3. Use high-resolution graphics
4. Format your graphics and include labels, legends, and captions

Section 2

RMarkdown

Reproducible reports

The final product of a data analysis project is often a report: scientific publications, news articles, an analysis report for your company, or lecture notes for a class.

Now imagine after you are done you realize you:

- had the wrong dataset
- have a new dataset for the same analysis
- made a mistake and fix the error, or
- your boss or someone you are training wants to see the code and be able to reproduce the results

Situations like these are common for a data scientist.

R markdown

R markdown is a format for **literate programming** documents.

It is based on **markdown**, a markup language that is widely used to generate html pages.

You can learn more about markdown here:
https://www.markdowntutorial.com/

R markdown

Literate programming weaves instructions, documentation, and detailed comments in between machine executable code.

With R markdown, you need to **compile** the document into the final report.

You can start an R markdown document in RStudio by clicking on **File**, **New File**, then **R Markdown**. You will then be asked for a title and author.

Final reports can be to be in: HTML, PDF, Microsoft Word, or presentation formats.

R markdown

As a convention, we use the `.Rmd` suffix for these files.

Once you gain experience with R Markdown, you will be able to do this without the template and can simply start from a blank template.

In the template, you will see several things to note (in the following slides).

The Header

At the top you see:

```
---
title: "Nanostring Analysis"
author: "Evan Johnson"
date: "12/5/2019"
output: html_document:
---
```

One parameter that we will highlight is `output`. By changing this to, say, `pdf_document`, we can control the type of output that is produced.

The Header

```
---
title: "Nanostring Analysis"
author: "Evan Johnson"
date: "12/5/2019"
output:
  html_document:
    code_folding: hide
    toc: true
    toc_float: true
    theme: "flatly"
editor_options:
  chunk_output_type: console
---
```

R code chunks

In various places in the document, we see something like this:

````
```{r}
summary(pressure)
```
````

These are the code chunks. When you compile the document, the R code inside the chunk, in this case `summary(pressure)`, will be evaluated and the result included in that position in the final document.

R code chunks

To add your own R chunks, you can type the characters above quickly with the key binding command-option-I on the Mac and Ctrl-Alt-I on Windows.

This applies to plots as well; the plot will be placed in that position. We can write something like this:

````
```{r}
plot(pressure)
```
````

R code chunks

By default, the code will show up as well. To avoid having the code show up, you can use an argument. To avoid this, you can use the argument `echo=FALSE`. For example:

````
```{r, echo=FALSE}
summary(pressure)
```
````

R code chunks

Its a good habit to adding a label to the R code chunks. This will be very useful when debugging, among other situations. You do this by adding a descriptive word like this:

````
```{r pressure-summary}
summary(pressure)
```
````

Global options

One of the R chunks may contain a complex looking call:

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
````

knitR

We use the **knitR** package to compile R markdown documents. The specific function used to compile is the `knit` function, which takes a filename as input. RStudio provides a button that makes it easier to compile the document.

Other options

We will explore other R Markdown options: headers, tabsets, and latex equations in class and in your HW.

Note: From now on, all R-based homework will need to be turned in using an R Markdown document (uploaded to GitHub). Your document should include headers, descriptive text, R code, and plots/figures!

More on R markdown

There is a lot more you can do with R markdown. We highly recommend you continue learning as you gain more experience writing reports in R. There are many free resources on the internet including:

1. RStudio's tutorial: https://rmarkdown.rstudio.com
2. The cheat sheet: https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf
3. The knitR book: https://yihui.name/knitr/

Session info

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.31   lifecycle_1.0.3 magrittr_2.0.3  evaluate_0.19
##  [5] rlang_1.0.6     stringi_1.7.8   cli_3.5.0       rstudioapi_0.14
##  [9] vctrs_0.5.2     rmarkdown_2.19  tools_4.2.2     stringr_1.5.0
## [13] glue_1.6.2      xfun_0.36       yaml_2.3.6      fastmap_1.1.0
## [17] compiler_4.2.2  htmltools_0.5.4 knitr_1.41
```