Communicating Science using R Markown

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

2025-02-10

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

Reproducible Reports (R markdown)

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. Literate programming weaves instructions, documentation, equations, and detailed comments among executable code.

It is based on **markdown**, a markup language that is widely used to generate html pages. You can learn more about markdown with the following tutorial: **click here**

R markdown

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.

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.

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

Compiling the document using `knitR`

With R markdown, you need to **compile** the document into the final report. We use the `knitR` package to to do this. The specific function used to compile is the `knit` function, which takes a filename as input.

RStudio provides a `Knit` button that makes it easy and convenient to compile the document.

The Header

At the top of the document is the R markdown header:

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

R markdown reports can be to be in HTML, PDF, Microsoft Word, or presentation formats. By changing the `output` to, for example `pdf_document` or `word_document`, we can control the type of output that is produced.

The Header

Other output options include code folding, themes, etc.:

```
---
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.

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.

R Code Chunks

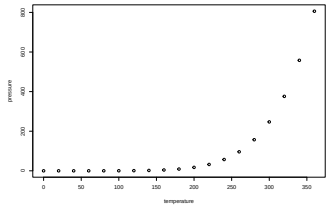
Its a good habit to adding a label to the R code chunks. This will be very useful when debugging, among other situations. You do this by adding a descriptive word like this:

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

R Code Chunks

We can also add plots to our report:

````
```{r, out.width = '30%', fig.align = 'center'}
plot(pressure)
```
````



The chunk options `out.width` and `fig.align` adjust the size and location

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)
```
````

If you want to include the code in the document, but not run the code and/or include results, you can use the argument `eval=FALSE`:

````
```{r, eval=FALSE}
# You can install the tidyverse using:
install.packages(tidyverse)
```
````

R Code Chunks (global `knitr` options)

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

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

The `include=FALSE` option in the chunk call will tell the system to run the R code, but not include it in the html/pdf/word report.

The R code in the chunk sets the `knitr` default to `echo=TRUE` in the call for any R chunks following this one.

Other Code Chunks

`knitr` can execute code in many languages besides R:

````
```{python}
x = [2, 1, 3]
print(x[0],"Hello Python!")
```
````

```
## 2 Hello Python!
```

Some of the available language engines include: Python, SQL, Bash, Rcpp, Stan, JavaScript, and CSS.

LaTeX Equations

One exciting feature about R markdown is that it allows you to include LaTeX equations to be rendered in html, pdf, or Word.

For example, to show the probability under the normal curve in the interval (a, b), you can use the LaTeX code:

```
\mbox{Pr}(a < x < b) =
  \int_a^b \frac{1}{\sqrt{2\pi}\sigma}
  e^{-\frac{1}{2}\left( \frac{x-\mu}{\sigma} \right)^2} \, dx
```

This will be rendered by `knitr` as:

$$\Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$$

Other Options (tabsets)

One very useful tool in Markdown is to add tabs to the html document:

```
## This is a header {.tabset}
### This is Tab 1
### This is Tab 2
## This is a new header (not in the tabs)
```

The `{.tabset}` option will make all lower level subheadings tabs in the html. The next same or higher level header will break out of the tabset.

Other Options (tabsets)

Pro tip: tabs can be generated dynamically:

````
## My Header {.tabset}
```{r, results = 'asis'}
n <- 10
for (i in 1:n){
  cat("###" , "Tab", i, "\n")
    print(i)
  cat("\n\n")
}
```
````

Other Options

We will explore other R markdown options: headers, tabsets, and LaTeX equations in class and in your Exercises.

Note: From now on, all R-based Exercises should preferably be done using an R markdown document. Your document should include headers, descriptive text, equations, 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/

More Practice

R markdown is a powerful tool for literate prgramming. To gain more practice, as part of your homework you will recreate the .Rmd file for the example file in the homework folder: "Rmarkdown_example.html."

Install the following packages:

For the next lecture on tidyverse R packages please install the following:

The tidyverse:

```
install.packages("tidyverse")
```

R Packages:

```
install.packages(c("devtools","styler","testthat"))
```

Shiny:

```
install.packages("shiny")
```

Session info

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: aarch64-apple-darwin20
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  LAPACK version 3.12.0
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: America/Denver
## tzcode source: internal
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] digest_0.6.37     fastmap_1.2.0     xfun_0.50         Matrix_1.7-2
##  [5] lattice_0.22-6    reticulate_1.40.0 knitr_1.49        htmltools_0.5.8.1
##  [9] png_0.1-8         rmarkdown_2.29    tinytex_0.54      cli_3.6.3
## [13] grid_4.4.2        compiler_4.4.2    rprojroot_2.0.4   here_1.0.1
## [17] rstudioapi_0.17.1 tools_4.4.2       evaluate_1.0.3    Rcpp_1.0.14
## [21] yaml_2.3.10       jsonlite_1.8.9    rlang_1.1.5
```