Literate Data Science with knitr and RMarkdown

ISA 616

Literate Data Science Programming

We have learned…

- it takes considerable effort to put data/results together in a sharable, reproducible way.
- most won’t take the time to document their work unless it is part of their normal workflow in developing the results.
- shared work producers and practitioners need to know: what was done, how it was done, why it was done, etc.

Problems

Before we had reproducibility tools…

- In the best scenarios, we used “copy/paste” to compile code, output, and intepretations/discussions in a readable document.
- In the next best scenarios, we saved documented code, output, and interpretations in separate files or locations.
- In most cases, we used point-and-click analyses blended with some coding, and our documentation, output, and interpretations were kept in separate files or locations.

Problems

This outdated approach…

- is based on the “just get it done as fast as possible” mindset
- leads to a lack of transparency and a lack of reproducibility
- leads to a lack of trust/confidence in **your skills** and **our profession**

Solutions

- Build a single “literate” document that “weaves” together code, output, and analysis together.
- Original idea comes from Don Knuth, a retired computer science professor from Stanford.
- Knuth’s philosophy, called “literate programming” is to allow programmers to write programs that can be read and understood by real people, not just machines.
- Knuth wrote the *Art of Computer Programming* in the 1980’s (it has been updated several times)
- Knuth wrote *Literate Programming* in 1980’s

A Literate Program

- Create a single document that has a stream of text and code.
- Document is divided into text and code chunks.
- Analysis code conducts the analysis.
- Presentation code formats the tables and graphs.
- The code/text chunks can be “woven” into readable documents.
- The code/text chunks can also be “tangled” into machine readable documents.

A Literate Program

- A Literate Program is a general concept. It requires
  - A documentation language
  - A programming language
- Example: The original **Sweave** system was developed by Friedrich Leisch and used LaTex and R.
- Example: **knitr** package that supports R code and a variety of documentation languages (Markdown, LaTex, AsciiDoc, HTML, etc.)

Some Basics in of RMarkdown

- `Markdown` is a language that converts plain text to HTML and other formats. It is also an R package that is a predecessor to `RMarkdown`.
- `RMarkdown` is an R package that converts a .Rmd file to a number of different file formats (HTML, doc, pdf, ppt, etc.).
- `pandoc` is an R package that is used by `RMarkdown` to accomplish some document conversions.
- `knitr` takes the plain text document with embedded code, executes the code and "knits" the results back into a document. For example, it converts an RMarkdown document, `.Rmd`, into a standard markdown file, `.md`, or an `.Rhtml` into a `.html`.
- When you "knit" a .Rmd file:

`.Rmd` -> `knitr` -> `.md` -> `pandoc` -> `desired format` (html, doc, pdf, etc.)

Markdown vs. Markup?

- Markdown is a simplified version of a text editor known as a “Markup” language.
- LaTex and HTML are Markup languages, where you add “tags” to enhance regular text.
  - These languages are not WYSIWYG editors. The tags make them difficult to read.
- Markdown is a simplified version that is easier to read with only a few formatting elements.

So you want to make your work Reproducible?

- Decide to do it (from the start)!
- Keep track of things, perhaps with a version control system.
- Use software that can be coded.
- Save code, not output when possible.

“Pros” of Reproducible Analysis…

- Text and code all in one place, in a logical order
- Data and results automatically update to reflect changes
- Code is live-if it doesn’t compile, you can fix it.

“Cons” of Reproducible Analysis…

- Text and code all in one place…if there are huge chunks of code the document is hard to read.
  - Code in small chunks
  - hide/show some code chunks
  - annotate! annotate! annotate!
- Documents are slow to produce.
- Documents can be VERY slow to run/render.

What are knitr/RMarkdown documents good for?

- Homework that requires code
- Creating slides that include code/output
- Tutorials
- Software manuals
- Reports that require repeated analyses
- Data processing, analysis, and summaries

What are knitr/RMarkdown documents not good for?

- Long research articles
- Documents that require precise formatting
- Analyses that require time consuming computations (e.g. parallel computing)

An Example and Documentation

- Here is an example of what I think is a *good* documentation of an analysis: https://fmegahed.github.io/fatigue_case_jqt.html
- Here is a *free* introductory book to help you format your documents: https://bookdown.org/yihui/rmarkdown/
- Here is a *more advanced* book to help you format your documents: https://bookdown.org/yihui/rmarkdown-cookbook/

What to include in your Analysis Document

1. An overview of the purpose of your analysis including details or references necessary.
2. A description of your data:
   - the source
   - the variables with definitions or a link to a codebook
   - the number of observations in your data set
   - detail on missingness
   - a glimpse of your data if possible (e.g. head and tail)

What to include in your Analysis Document

3. Details on data preprocessing:
    - Feature generation
    - Imputation
    - Cleaning or merging of categories
    - Outlier removal
    - Anything that changes your data from the original form

What to include in your Analysis Document

4. Your **Final** Analysis in small pieces with annotation
5. Graphs to visualize different steps in your analysis
6. Clear discussion of why you made analysis choices
7. References to papers or citations you used to make decisions about the analysis

What to include in your Analysis Document

8. The code used to generate your final results
9. A discussion of your results and conclusions

What NOT to include in your Analysis Document

1. A printout of all of your data
2. A summary dump that is poorly formatted or hard to follow
3. Undocumented output or results
4. Everything you tried that didn't work

What *NOT* to include in your Analysis Document

5. A narrative of everything you tried. Here is an example that I commonly see…

First I loaded my data into R. Next I printed the first six rows of the data using the head function. Next I used the summary function to see the descriptive statistics. After that I decided to run a regression between Y and several X variables. When I tried to run the lm function, I found that there was a lot of missing data, so I went back and tried to figure out which variables had missing values. I noticed that X2 had all of the missing values, so I imputed those with the mean. After I did that, I reran the lm function at got the following results…

Most Often, Your Analysis Document Should Be

Results Driven with clear documentation on the Process

NOT

Process Driven with clear documentation on the Results

A Simple Self-Check

- Are the results believable?
- Are the results true?
- Are the results justified?