Explore a working knitr document

2. Bring it all together

- Contrast literate programming with stand-alone scripts and stand-alone documentation.
- Relate literate programming to reproducible generation of scientific figures and analysis results.
- Recognize benefits of and opportunities for applying literate programming.
- Produce and modify R knitr reports.
- Identify intermediary files, and thus which files can be regenerated. {: .objectives}

Interactive knitr document: countryPick4.Rmd.

Introduction

Reading code is often difficult, even if you are the programmer. Literate programming is an approach to writing code that combines plain English interspersed with code to help explain the why and how behind the code. The end result is often a report that provides narrative to your work. These reports alleviate many of the problems associating with computational work including:

1. Making your code pleasurable to read.
2. Reduction of tedium and manual processes when making changes to experiments.
3. Reduction of friction for communication of results.
4. Begins the process of writing papers as the data, figures and rationale for each project are all in one place and can be copied and edited when writing your papers.

Essentially literate programming acts as your lab notebook for your computational work. As with scientific notebooks, there are many styles to writing in one. Every project is unique, but the end goal is always to make it as easy as possible for you and others to reproduce your work.

In this lesson we are going to use `RStudio` to combine `R` code, markdown, and knitr to create reports that can be saved in many formats for distribution. Depending on the language you can combine the same or similar tools for creating literate programming reports, for example for `Python` reports, `IPython` and `Project Jupyter` are powerful options.

Step 1: Open .Rmd Document and Run Code

Open countryPick4.Rmd. This is a report exploring population size of four countries.

Save this document with a new filename. The filename should end in `.Rmd`. Use what you have learned from file naming to give an appropriate new name to this report.

> ##Challenge 1 - Modify the report. 1. Re-run the report, but modify the code to choose four new countries. 2. (Bonus) If you are familiar with `R` and `ggplot`, try adding a fifth country to this report.

Step 2: Save new report and knit

Click on "Knit HTML" or do `File > Knit`. `RStudio` should display a preview of the resulting HTML. Also look at the file browser (which should be pointed at the directory where you saved the .Rmd file). You should see the `RMarkdown` document, i.e. `foo.Rmd` and the resulting HTML `foo.html`.

- Distinguish what was the input and output in creating this document. What role did the `.Rmd` file take and what the .html file?
- Which file can you delete while still being able to fully reproduce the result? What are all the files that are needed to re-create the `.html` file?

Step 3: Develop a new report

We'll use the code chunk below to develop a new report. It should contain a title, two text blocks and two code blocks. The code requires the dplyr and babynames packages. We'll use the RStudio package pane to install them.

```
library(dplyr)
library(babynames)

# Plot name frequency over time
babynames %>%
  filter(name == "Mary" & sex == "F") %>%
  ggplot(aes(x = year, y = n)) +
  geom_line() + labs(x = "Year", y = "Number Born",
                     title = "Mary as a Girl-Name")

# Compare name use by gender over time
babynames %>%
  filter(name == "Leslie") %>%
  ggplot(aes(x = year, y = n)) +
  geom_line(aes(color = sex)) + labs(x = "Year", y = "Number Born",
                                     title = "Leslie, by Sex")
```

Sometimes left-over objects in the workspace make a report appear to fail even though it is correct. Clean out your workspace, restart `R`, and re-run everything if you see weird failures. There are lots of chunk menu items and keyboard shortcuts to accelerate this workflow. Render the whole document often to catch errors when they're easy to pinpoint and fix. Save often every time you reach a point that you'd like as a "fall back" position.

Step 4: Publishing your report

Although not required, one of most accessible and often easiest ways to share `RMarkdown` reports is over the web. Rpubs is a free database of `R knitr` documents, and makes it very easy to publish your knitr documents, especially when using `RStudio`. In RStudio, create a new R Markdown document by choosing `File > New > RMarkdown`. Click the `Knit HTML` button in the doc toolbar to preview your document. In the preview window, click the Publish button. Browse recently published reports at Rpubs.

Another popular option is Github, which allows the publication and hosting of .md files for free. Github also provides the power of version control through Git.

Troubleshooting

Make sure `RStudio` and the rmarkdown package (and its dependencies) are up-to-date. In case of catastrophic failure to render `RMarkdown`, consider that your software may be too old. `RMarkdown` has been developing rapidly (written 2016-12), so you need a very current version of `RStudio` and `RMarkdown`.

You may need to get rid of your `.Rprofile`, at least temporarily. Specifically, if you've got anything in there relating to `knitr`, `markdown`, `RMarkdown` and `RStudio` stuff, it may be preventing the installation or usage of the most recent versions.

Insert a chunk in your `.Rmd` document so that it renders even when there are errors. Some errors are easier to diagnose if you can execute specific `R` statements during rendering and leave more evidence behind for forensic examination. Put this chunk:

near the top of your R Markdown document if you want to the report to run in full even if there are errors, i.e. turn `foo.Rmd` into `foo.md` and/or `foo.html` no matter what.

Check whether your working directory is set correctly. Drop these commands into `R` chunks to check the above: `getwd()` will display working directory at run time.

Don't try to change working directory within an `RMarkdown` document. And don't be in a hurry to create a complicated sub-directory structure. `RStudio/knitr/RMarkdown` (which bring you the "Knit HTML" button) are rather opinionated about the working directory being set to the `.Rmd` file's location and about all files living together in one big happy directory.

Resources:

- Knitr Cheat Sheet - cheat sheet for `knitr` code.
- RStudio - Up-to-date RStudio versions also contain direct links to these "cheatseets" `Help > Cheatsheets`
- Ipython - Powerful option for writing literate programming in `Python` .
- Project Jupyter - The offsrping of Ipython allows literate programming in a variety of languagues, including `R` and `julia`.

Credits This material was adapted from an activity by Jenny Bryan.