30 R Markdown workflow

Earlier, we discussed a basic workflow for capturing your R code where you work interactively in the *console*, then capture what works in the *script editor*. R Markdown brings together the console and the script editor, blurring the lines between interactive exploration and long-term code capture. You can rapidly iterate within a chunk, editing and re-executing with Cmd/Ctrl + Shift + Enter. When you're happy, you move on and start a new chunk.

R Markdown is also important because it so tightly integrates prose and code. This makes it a great **analysis notebook** because it lets you develop code and record your thoughts. An analysis notebook shares many of the same goals as a classic lab notebook in the physical sciences. It:

- Records what you did and why you did it. Regardless of how great your memory is, if you don't record what you do, there will come a time when you have forgotten important details. Write them down so you don't forget!
- Supports rigorous thinking. You are more likely to come up with a strong analysis if you record your thoughts as you go, and continue to reflect on them. This also saves you time when you eventually write up your analysis to share with others.
- Helps others understand your work. It is rare to do data analysis by yourself, and you'll often be working as part of a team. A lab notebook helps you share not only what you've done, but why you did it with your colleagues or lab mates.

Much of the good advice about using lab notebooks effectively can also be translated to analysis notebooks. I've drawn on my own experiences and Colin Purrington's advice on lab notebooks (http://colinpurrington.com/tips/lab-notebooks) to come up with the following tips:

- Ensure each notebook has a descriptive title, an evocative filename, and a first paragraph that briefly describes the aims of the analysis.
- Use the YAML header date field to record the date you started working on the notebook:

  ```
  date: 2016-08-23
  ```

  Use ISO8601 YYYY-MM-DD format so that's there no ambiguity. Use it even if you don't normally write dates that way!
- If you spend a lot of time on an analysis idea and it turns out to be a dead end, don't delete it! Write up a brief note about why it failed and leave it in the notebook. That will help you avoid going down the same dead end when you come back to the analysis in the future.
- Generally, you're better off doing data entry outside of R. But if you do need to record a small snippet of data, clearly lay it out using `tibble::tribble()`.
- If you discover an error in a data file, never modify it directly, but instead write code to correct the value. Explain why you made the fix.
- Before you finish for the day, make sure you can knit the notebook (if you're using caching, make sure to clear the caches). That will let you fix any problems while the code is still fresh in your mind.

- If you want your code to be reproducible in the long-run (i.e. so you can come back to run it next month or next year), you'll need to track the versions of the packages that your code uses. A rigorous approach is to use **packrat**, http://rstudio.github.io/packrat/, which store packages in your project directory, or **checkpoint**, https://github.com/RevolutionAnalytics/checkpoint, which will reinstall packages available on a specified date. A quick and dirty hack is to include a chunk that runs `sessionInfo()` — that won't you let easily recreate your packages as they are today, but at least you'll know what they were.

- You are going to create many, many, many analysis notebooks over the course of your career. How are you going to organise them so you can find them again in the future? I recommend storing them in individual projects, and coming up with a good naming scheme.