Sharing Analyses Across Research Groups

Hershel Mehta

June 6, 2018

Executive Summary

In the last four years, I have worked in a lab researching affective neuroscience and behavioral economics. Our lab is interesting because we are technically in the psychology department, but have students and collaborators from disciplines as diverse as economics, human biology, bioengineering, and machine learning. As such, we do a lot of work learning the languages of other disciplines and learning how to integrate our analyses into those of our collaborators. This process involves several months of meetings and reading to ensure that we understand our collaborators' workflows and data-generating process. What we have desired for many years is a set of standardized conventions that help us and our collaborators share our analyses across multiple research groups.

A few of the central questions I have had when considering how to construct a collaborative, shareable workflow with multiple research groups are,

1) What are the best practices for sharing data?
2) What are the best practices for strucuturing analyses?
3) What would an implementation of these conventions look like in practice?

This annotated bibliography presents my attempt at answering these questions. In fact, each of my top three articles tackles one of these questions. Below, I summarize a few of my main conclusions from these readings:

- Think about where each part of your analysis belongs in your project directory: In other words, the way that you package your analysis matters. For instance, most sources agree on the principle that "data is immutable", which essentially means that you should start from "raw data" and build your analyses from there. There are several conflicting opinions for where your data processing code should go. All processing scripts belong in a source code folder; however, that does not mean that all of your code belongs in a source code directory. Indeed, analytical code lends itself more to notebook files (e.g., R Markdown or Jupyter Notebooks) becuase you can interleave text and code to both create, report, and easily reproduce analytical results. Generally speaking, having a lot of separate files that do one thing really well is much better than having a heap of code in one massive file.
- Use as many existing tools as possible to automate your workflow: All of the implementations I have studied that do a good job reproducing results attempt to automate as much of their workflow as possible. The problem I see with many ad-hoc solutions is that they are not part of what I call an "ecosystem" of packages that all work together. The benefit of both R Packages (for packaging R analyses) and `cookiecutter` (for packaging Python analyses) is that they easily connect to various tools that allow for other software engineering best practices, such as unit testing and documentation.
- Standardize data: when collaborating with different labs, a big problem is in understanding the data itself, especially if your lab group was not the group that collected the data. To reduce that frustration, explicitly define the data encoding practices and other conventions (e.g., creating a data dictionary) to ensure that all teams are on the same page when it comes to data.

Taken together, I believe that these practices will help me establish reasonable guidelines for collaborations from a wide diversity of research groups. If I can clearly establish what is expected in terms of delivering data and results, I believe I can collaborate on projects more quickly while keeping reproducibility and collaboration at the center of my workflow.

Top Three

Ellis & Leek (2017)

Question: What are the best practices for sharing data?

In my experience as a psychology researcher, I have found that several problems have arisen when collaborating with other groups that have collected the data. Often times, data encoding schemes are very particular to each lab (e.g., using numbers to represent gender in different ways), which makes it much more difficult to conduct analyses without asking a lot of clarifying questions about the nature of the data. Moreover, this style of collaboration makes it difficult to merge together my results with the other lab's results, which makes reproducibility even more challenging than it should be. Ideally, a schema would exist for how to hand off data so each team can move forward as independentally as possible while still retaining reproducibility.

Ellis & Leek (2017) lay out a roadmap for the data-generating team and external data analysts so that collaboration can occur as seamlessly as possible. They argue that the data-generating team should be responsibile for leaving the "raw data" untouched, for creating "tidy data" with a script (ideally), and for writing a data dictionary so that each variable is well-defined. They also specify conventions for naming variables (e.g., avoid spaces in variable names) and encoding data (e.g., use male / female instead of 1 / 0) to minimize the amount of clarifying questions required to comprehend the data. In particular, I found Figure 2 very helpful in explicitly articulating the "what" and "why" behind each convention the authors found relevant when sharing data with multiple teams. As the authors note in their conclusion, these practices "dramatically decreases the workload on the statistician and minimizes the likelihood of errors during analysis. By taking the time to tidy the data, the data generator, who knows the details of the data generated better than anyone else, can expect to get the analysis back sooner and can be more confident in its accuracy."

This article made me realize that we often talk about best practices in the context of sharing code, but very rarely talk about the conventions for sharing data. I thought that the authors effectively fill that gap.

Marwick et al. (2017)

Question: What are the best practices for strucuturing analyses?

When considering how to structure my projects so that they are collaborative, I found that there were very few comprehensive resources on best practices. I often found myself on forums, trying to interpret a few users workflows and going in circles. I believe this article by Marwick et al. (2018) fills a much needed gap in the literature by defining a "research compendium": a reproducible file structure that separates data, method, and output. Specifically, this article argues that users should build "research compendia" using R's excellent package ecosystem (e.g., `roxygen2` and `usethis`).

The article argues that this package ecoystem is ideal for reproducible and collaborative research becuase of its built-in "quality control mechanisms". For instance, instead of writing functions in many directories of our project, we can put all of our functions in the "R/" directory, and gain the benefit of the surrounding pacakge ecosystem (e.g., `testthat`).

Another unique aspect of this article is that it mentions different collaborative structures for different sizes of projects. In my class, for instance, which just requires small data and a few basic plots and analayses, a single .Rmd file might be good enough for reproducibility and collaboration. However, as your project grows, putting everything inside of a single .Rmd file becomes hard to maintain and hard to share to a colleague to collaborate on. In separating our code and analyses into separate folders, we ensure that we follow another best practice we have discussed in this class, which is to ensure that each piece of our project "does one thing, and does it well" (i.e., the Unix Philosophy). Instead of having functions, visualizations, and reports in a single file, we will now have a better infrastructure for ensuring that the Unix Philosophy holds in all of our analyses.

DrivenData (n.d.)

Question: What would an implementation of these conventions look like in practice?

I thought that this article by DrivenData did a great job at introducing `cookiecutter`, a Python package that implements many of the practices I have mentioned above for collaboration and many of the other practices we have spoken about in this class. With over 7000 stars on GitHub and 178 contributors, `cookiecutter` seems to have one of the broadest reproducibility communities in data science.

At first glance, what makes `cookiecutter` special is its standardized but lightweight directory structure. For instance, `cookiecutter` includes a place for all exploratory notebooks in a `notebooks` directory and data dictionaries in a `references` directory. Although these conventions do indeed give clear guidelines about how to build up a reproducible and collaborative analysis, DrivenData (n.d.) go beyond those surface-level considerations and explicitly define the "Opinions" that underlie the decisions in `cookiecutter`.

A couple of the "Opinions" that resonated with me were,

- "Notebooks are for exploration and communication" - In my data science career, I have been using notebooks for nearly everything, from preparing my data to conducting analyses. Here, the authors argue that notebooks are best served for a certain purpose: exploring your data and communicating results. They note that notebooks are more difficult for collaboration (e.g., since the structure of notebooks make merging difficult), suggesting that you should use scripts for these purposes. I liked this opinion because I now understand what type of code should be put in what type of file.
- "Analysis is a DAG" - Although I have heard in the past that I should use `make`, I have never understood why. This article's framing of analysis as a DAG (i.e., a linear set of dependencies) helped me see that `make` is a diret implementation of this principle. In other words, `make` explicitly defines the dependencies for each analysis and updates only those dependencies whose inputs have changed in some way. In doing so, it serves not only as a tool for reproducibility but also as useful documentation for the dependencies.

As an active R user, I found this article's take on `cookiecutter` useful in understanding a totally different way at implementing a reproducible workflow. I hope to see many of these principles implemented inside of the R Packages ecosystem as well.

Annotated Bibliography

Baumer, B. (2017). Lessons From Between the White Lines for Isolated Data Scientists, The American Statistician, 72:1, 66-71, DOI: 10.1080/00031305.2017.1375985. Available at, https://www.tandfonline.com/doi/full/10.1080/00031305.2017.1375985?src=recsys

Broman, K. (n.d.). Organize your data and code. Available at, http://kbroman.org/steps2rr/pages/organize.html

Boettiger, C. (2012). My research workflow, based on Github. Available at, http://www.carlboettiger.info/2012/05/06/research-workflow.html

Bostock, M. (2013). Why Use Make. Available at, https://bost.ocks.org/mike/make/

Bryan, J. (2017). Project-oriented workflow. Available at, https://www.tidyverse.org/articles/2017/12/workflow-vs-script/

Diaz, M. (2013). Organizing the project directory. Available at, https://nicercode.github.io/blog/2013-05-17-organising-my-project/

DrivenData (n.d.). Cookiecutter Data Science: a logical, reasonably standardized, but flexible project structure for doing and sharing data science work. Available at, https://drivendata.github.io/cookiecutter-data-science/.

Ellis, Shannon E. & Leek, Jeffrey T. (2017). How to Share Data for Collaboration, The American Statistician, 72:1, 53-57, DOI: 10.1080/00031305.2017.1375987. Available at, https://www-tandfonline-com.stanford.idm.oclc.org/doi/full/10.1080/00031305.2017.1375987?src=recsys

Fitzjohn, R. (2013). Designing projects. Available at, https://nicercode.github.io/blog/2013-04-05-projects/

Greenfield, D. (2013). Cookiecutter: Project Templates Made Easy. Available at, https://www.pydanny.com/cookie-project-templates-made-easy.html

Knupp, J. (2013). Open Sourcing a Python Project the Right Way. Available at, https://jeffknupp.com/blog/2013/08/16/open-sourcing-a-python-project-the-right-way/

Marwick, B., Boettiger, C., & Mullen, L. (2017). Packaging Data Analytical Work Reproducibly Using R (and Friends), The American Statistician, 72:1, 80-88, DOI: 10.1080/00031305.2017.1375986. Available at, https://www-tandfonline-com.stanford.idm.oclc.org/doi/full/10.1080/00031305.2017.1375986?scroll=top&needAccess=true

Noble, W. S. (2009). A Quick Guide to Organizing Computational Biology Projects, PLoS Comput Biol 5(7): e1000424. https://doi.org/10.1371/journal.pcbi.1000424

Wickham, H. (2017). Research compendia. Note prepared for the 2017 rOpenSci Unconf. Available at, https://docs.google.com/document/d/1LzZKS44y4OEJa4Azg5reGToNAZL0e0HSUwxamNY7E-Y/edit#