Nice R Code

Punning code better since 2013

- RSS

Search

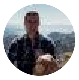
Navigate… ▼

- Blog
- Archives
- Guides
- Modules
- About

Designing projects

Posted by Rich FitzJohn - 05 April 2013 -

The scientific process is naturally incremental, and many projects start life as random notes, some code, then a manuscript, and eventually everything is a bit mixed together.



Vince Buffalo
@vsbuffalo

Managing your projects in a reproducible fashion doesn't just make your science reproducible, it makes your life easier.

12:56 AM - Apr 15, 2013

38 35 people are talking about this

Directory layout

A good project layout helps ensure the

- Integrity of data
- Portability of the project
- Easier to pick the project back up after a break

There is no one way to lay a project out. Daniel and I both have different approaches for different projects, reflecting the history of the project, who else is collaborating on that project.

Here are a couple of different ideas for laying a project out. This is the basic structure that I tend to use:

```
proj/
├── R/
├── data/
├── doc/
├── figs/
└── output/
```

- The `R` directory contains various files with function definitions (but *only* function definitions - no code that actually runs).
- The `data` directory contains data used in the analysis. This is treated as *read only*; in paricular the R files are never allowed to write to the files in here. Depending on the project, these might be csv files, a database, and the directory itself may have subdirectories.
- The `doc` directory contains the paper. I work in LaTeX which is nice because it can pick up figures directly made by R. Markdown can do the same and is starting to get traction among biologists. With Word you'll have to paste them in yourself as the figures update.
- The `figs` directory contains the figures. This directory *only contains generated files*; that is, I should always be able to delete the contents and regenerate them.
- The `output` directory contains simuation output, processed datasets, logs, or other processed things.

In this set up, I usually have the R script files that *do* things in the project root:

```
proj/
├── R/
├── data/
├── doc/
├── figs/
├── output/
└── analysis.R
```

For very simple projects, you might drop the R directory, perhaps replacing it with a single file `analysis-functions.R` which you source.

The top of the analysis file usually looks something like

```
library(some_package)
library(some_other_package)
source("R/functions.R")
source("R/utilities.R")
```

…followed by the code that loads the data, cleans it up, runs the analysis and generates the figures.

Other people have other ideas

- Carl Boettiger is an open science advocate who has described his layout in detail. This layout uses R packages for most of the code organisation, and would be a nice approach for large projects.
- This article in PLOS Computational Biology describes a general framework.

Treat data as read only

In my mind, this is probably the most important goal of setting up a project. Data are typically time consuming and/or expensive to collect. Working with them interactively (e.g., in Excel) where they can be modified means you are never sure of where the data came from, or how they have been modified. My suggestion is to put your data into the `data` directory and treat it as *read only*. Within your scripts you might generate derived data sets either temporarily (in an R session only) or semi-permanantly (as an file in `output/`), but the original data is always left in an untouched state.

Treat generated output as disposable

In this approach, files in directories `figs/` and `output/` are all generated by the scripts. A nice thing about this approach is that if the filenames of generated files change (e.g, changing from `phylogeny.pdf` to `mammal-phylogeny.pdf`) files with the old names may still stick around, but because they're in this directory you know you can always delete them. Before submitting a paper, I will go through and delete all the generated files and rerun the analysis to make sure that I can create all the analyses and figures from the data.

Separate function definition and application

When your project is new and shiny, the script file usually contains many lines of directly executated code. As it matures, reusable chunks get pulled into their own functions. The actual analysis scripts then become relatively short, and use the functions defined in scripts in `R`. Those scripts do nothing but define functions so that they can always be `source()`'d by the analysis scripts.

Setting up a project in RStudio

This gets rid of the #1 problem with most people's projects face; where do you find the data. Two solutions people generally come up with are:

1. Hard code the full filename for each file you load (e.g., `/Users/rich/Documents/Projects/Thesis/chapter2/data/mydata.csv`)
2. Set the working directory at the beginning of your script file `/Users/rich/Documents/Projects/Thesis/chapter2` then doing `read.csv("data/mydata.csv")`

The second of these is probably preferable to the first, because the "special case" part is restricted to just one line in your file. However, the project is still now quite fragile, because moving it from one place to another, you must change this file. Some examples of when you might do this:

- Archiving a project (moving it from a "current projects" directory to a new projects directory)
- Giving the code to somebody else (your labmate, collaborator, supervisor)
- Uploading the code with your manuscript submission for review, or to Dryad after acceptance.
- New computer and new directory layout (especially changing platforms, or if your previous mess got too bad and you wanted to clean up).
- Any number of new reasons

The second case hints at a solution too; if we can start R in a particular directory then we can just use paths *relative to the project root* and have everything work nicely.

To create a project in R studio:

- "Project": "Create Project…"
- choose "New Project, (start a project in a new directory)".
- Leave the "Type" as the default.
- In the "Directory name" type the name for the project. This might be `chapter2` for a thesis, or something more descriptive like `fish_behaviour`.
- In the "Create project as a subdirectory of" field select (type or browse) for the parent directory of the project. By default this is probably your home directory, but you might prefer your Documents folder (I have mine in `~/Documents/Projects`).

Posted by Rich FitzJohn 05 April 2013

Tweet

« Plans for 'Nice R code module', Macquarie University 2013 Why I want to write nice R code »

Comments

Recent Posts

- Figure functions
- Modifying data with lookup tables
- Organizing the project directory
- How long is a function?
- Excel and line endings