Robin's Blog

Search for:

Search

Categories

Academic (83)
GIS (33)
Publications (4)
Remote Sensing (48)
Computing (42)
Computer Help (10)
Linux (10)
OSX (13)
Windows (11)
Disability (1)
Essays (2)
Home Automation (5)
How To (28)
Humour (3)
Lessons learnt from… (2)
Meta (2)
My Software (13)
bib2coins (1)
Py6S (9)
Programming (94)
C (4)
IDL (8)
LaTeX (5)
Previously Unpublicised Code (6)
Python (62)
R (11)
Reviews (15)
Books (12)
Technology (3)
Short Interesting Things (5)
TOK-related (5)

In praise of ProjectTemplate for reproducible research

September 7, 2012

As you might know from some of my previous posts, I'm a big fan of making my scientific work **reproducible**. My main reasons for being so keen on this are:

1. Reproducibility is key to science – if it can't be reproduced then it can not be verified (that is, the experiment can't be tried again to determine if the same result was produced then no-one can verify your work, and no-one can falsify it if it was incorrect), and therefore (according to various scientific philosophers such as Popper) it isn't science.

2. It's actually really useful for me as a researcher. Have you ever come back to a project six months after stopping work on it (possibly because you'd submitted a paper on it, and had to wait *ages* for the reviewers comments) and found it almost impossible to work out how to produce a certain graph or table, or which data was used to produce a certain result? Making your work reproducible by other scientists also means it will be reproducible by you when you've forgotten all about how it worked!

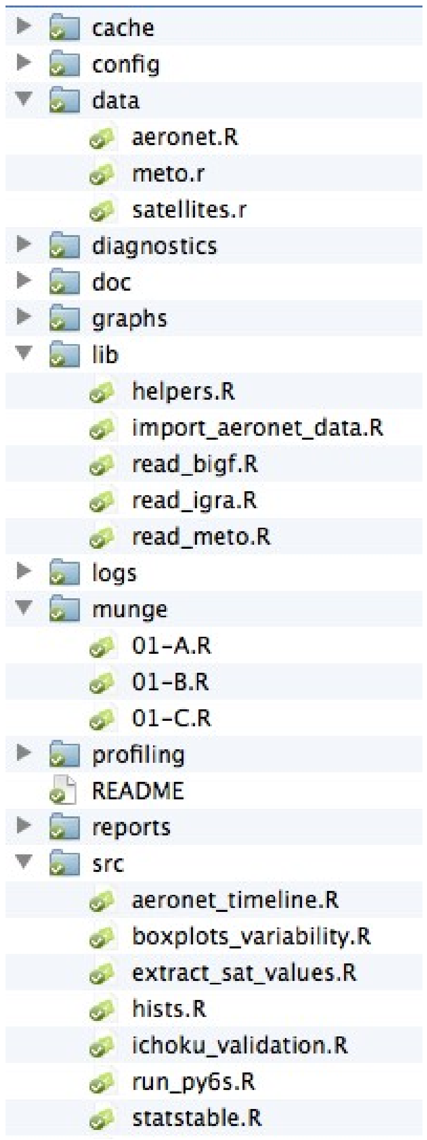
Basically reproducibility in scientific research these days means code. You could write a long Word document saying exactly how you processed all of your data (good luck keeping it up-to-date) and then run through all of that again, but in most of my work I use code in a variety of languages (Python, R and IDL mostly) to do the processing for me.

The beauty of this (aside from not spending ages clicking around in frustrating dialog boxes) is that doing your research through code gets it a long way to being reproducible without any other work on your part. You created your original outputs through code, so you can reproduce them just by running the code again! Simple isn't it?

Well, unfortunately, it's not quite that simple. Do you know which exact bit of data you used to create that code? Did you pre-process the data before using it in your code? Did you some processing on the data that the code produced before putting into a table/graph in your paper? Will you remember these things in six months/six years if you need to reproduce that bit of work yourself (or, more scarily, if someone emails you to tell you that they think your results were wrong…)? Unfortunately, I think that's unlikely.

Anyway, to get to the point of this post: I have recently been using a R package called ProjectTemplate which has really helped me make my research properly reproducible. This package generates a standard folder structure for your R code, and provides a range of useful functionality for automatically loading data and caching the results of computations. I've been using this for a report that I wrote recently for my PhD supervisors (something which may turn into a paper sometime – hopefully), and it's been great.

I'm not going to give a full overview of all of the functionality of ProjectTemplate here, but I'll show you a bit about how I use it. Firstly, here is my folder structure for this project:



```
▶ cache
▶ config
▼ data
    aeronet.R
    meto.r
    satellites.r
▶ diagnostics
▶ doc
▶ graphs
▼ lib
    helpers.R
    import_aeronet_data.R
    read_bigf.R
    read_igra.R
    read_meto.R
▶ logs
▼ munge
    01-A.R
    01-B.R
    01-C.R
▶ profiling
  README
▶ reports
▼ src
    aeronet_timeline.R
    boxplots_variability.R
    extract_sat_values.R
    hists.R
    ichoku_validation.R
    run_py6s.R
    statstable.R
```

Folder structure for my ProjectTemplate project

As you can see there are quite a few folders here with code in:

- **data:** scripts for loading in data (or data files themselves, which will be loaded automatically if they are of certain file types)
- **lib:** functions that will be used throughout the project
- **munge:** scripts to manipulate or 'munge' before use in the rest of the project (ie. pre-processing)
- **src:** scripts used to actually do the processing and analysis of the data

There are also quite a few folders that I use that I haven't shown expanded above:

- **cache:** stores cached R objects (eg. results from time-consuming processing steps)
- **graph:** stores graphical outputs from the analysis

So, what we've got here is a place to put re-usable functions (basically functions that could eventually go into a separate R package – eg. for reading specific format data files etc), a place to put pre-processing code, and a place to put actually scientific analysis code. You'll see there are loads of other folders that I haven't mentioned that I don't really use, but I suspect I will probably use them in new projects in the future.

The beauty of this folder structure is that the folder that contains the structure above can be simply zipped up, given to someone else and then they can run it themselves. How do they do that? Simple, change the working directory to the folder and run:

Uncategorized (1)

Recent posts

Regression in Python using R-style formula – it's easy!
August 20, 2016

Reminder about cross-platform case-sensitivity differences
August 11, 2016

'Got multiple values for argument' error with keyword arguments in Python classes
August 3, 2016

Showing code changes when teaching
July 25, 2016

Reading AERONET data in Pandas: a simple helper function
July 17, 2016

Subscribe

Subscribe via RSS
Subscribe via email

```
library(ProjectTemplate)
load.project()
```

That will load all of the library needed, load the data (from files or cache), pre-process it (or load the results from the cache) and get it to the stage where you can run any of the files in `src`. Great!

The brilliant thing is that each of my scripts in the src folder will produce one or two outputs for my report. All of my graphs are saved as PDFs into the graphs folder, ready to be included directly into a LaTeX document, and my tables are produced as R data frames and then converted to LaTeX tables using the xtable package.

So, what's the practical upshot of this? Well, if I come back to this in six months I can run any analysis that I use for my report by typing a couple of lines of code and then running the relevant file. It also meant that when I realised half-way through my writing up that I had accidentally done all of my results (about 5 graphs, and 4 big tables) based on some incorrect input data (basically I had data for half of Europe rather than just the UK, which makes a big difference to the ranges of atmospheric data!) it took me about 30 minutes to generate all of the new results by simply changing a line of code where the data was imported, running the pre-processing again (which took about 20 minutes of the 30 minutes time!) and then running each file to generate the required graph PDF files or xtable LaTeX code.

Hopefully this will have made sense to you all – stay tuned for some more reproducible research posts in the near future.

Categorised as: Academic, GIS, Programming, R, Remote Sensing

6 Comments

John Minter says:
September 8, 2012 at 2:24 pm

Thanks for the thought-provoking post.
I work as a specialist in microscopy and image processing and have been on a similar journey to make my workflow more reproducible. I use a much simpler folder tree: a project folder containing data, R, TeX, C, and script folders. I use a sensible .gitignore file to keep required files under source control. The script folder also contains a "rule-them-all" script that will regenerate the analysis with a single click.
My major concern these days is how to balance the reproducible research idea of having the project self-contained with the programming practice of "don't repeat yourself" (DRY). I tend to use a "global" tree for