Scientific Computing Best Practices

Luke Chang
Dartmouth College
MIND Summer School 2017

Goals: Maximizing Efficiency and Minimizing Errors

How can we get our work done as quickly as possible?

How can we minimize the number of errors and bugs in our data analysis?

http://journal.frontiersin.org/article/10.3389/fpsyg.2014.01435/full

American Economic Review: Papers & Proceedings 100 (May 2010): 573–578
http://www.aeaweb.org/articles.php?doi=10.1257/aer.100.2.573

Growth in a Time of Debt

By Carmen M. Reinhart and Kenneth S. Rogoff*

Does High Public Debt Consistently Stifle Economic Growth? A Critique of Reinhart and Rogoff

Thomas Herndon* Michael Ash Robert Pollin

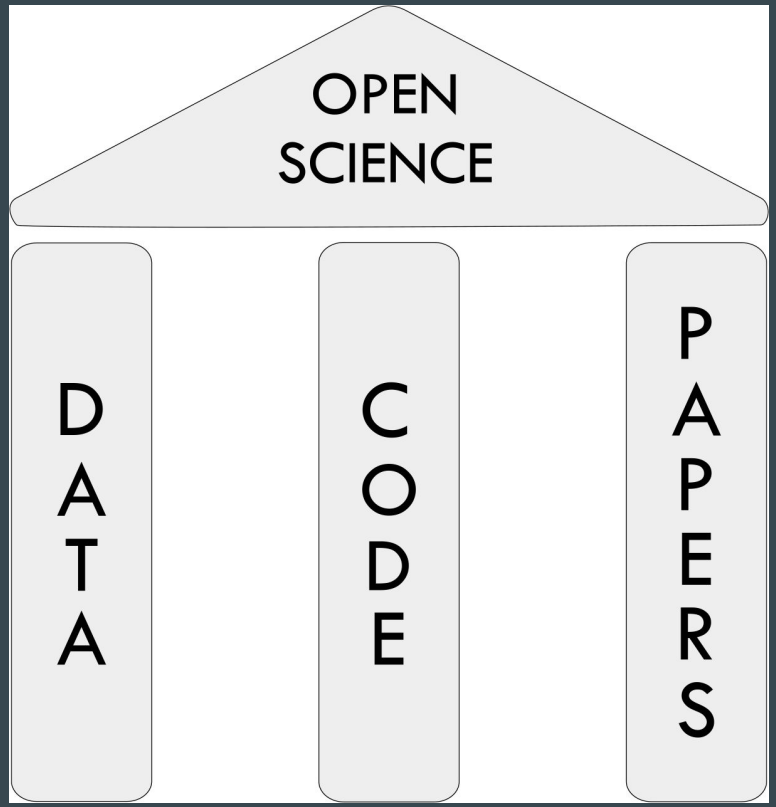
April 15, 2013

JEL codes: E60, E62, E65

Abstract

We replicate Reinhart and Rogoff (2010a and 2010b) and find that coding errors, selective exclusion of available data, and unconventional weighting of summary statistics lead to serious errors that inaccurately represent the relationship between public debt and GDP growth among 20 advanced economies in the post-war period. Our finding is that when properly calculated, the average real GDP growth rate for countries carrying a public-debt-to-GDP ratio of over 90 percent is actually 2.2 percent, not −0.1 percent as published in Reinhart and Rogoff. That is, contrary to RR, average GDP growth at public debt/GDP ratios over 90 percent is not dramatically different than when debt/GDP ratios are lower.

We also show how the relationship between public debt and GDP growth varies significantly by time period and country. Overall, the evidence we review contradicts Reinhart and Rogoff's claim to have identified an important stylized fact, that public debt loads greater than 90 percent of GDP consistently reduce GDP growth.



Gorgolewski & Poldrack, 2016 PLoS Biology

Towards Reproducible Workflows

We should strive to automateas much of our data preprocessing as possible

We should openly share our data. (https://openfmri.org/ , http://neurovault.org/ , personal website)

We should openly share our code used to analyze and generate paper figures (github, jupyter)

We should have consistent processing environments (Containers/VM)

Ideally this can all be done in the cloud (https://openneuro.org/,

Managing Code Projects

Version Control (Github)

Documentation

Tests

Tutorials

Automation with Github Webhooks

Webhooks are callbacks triggered by an event.

Github can trigger events upon a new commit or pull request

Testing via TravisCI - https://travis-ci.org

Code Coverage via Coveralls.io - https://coveralls.io/

Documentation via sphinx - http://www.sphinx-doc.org/en/stable/

Tutorials via sphinx-gallery - https://github.com/sphinx-gallery/sphinx-gallery

Example Github Webhoooks

TravisCI - https://travis-ci.org/ljchang/nltools/

Coveralls.io - https://coveralls.io/github/ljchang/nltools?branch=master

ReadTheDocs - http://neurolearn.readthedocs.io/en/latest/

Tutorials-http://neurolearn.readthedocs.io/en/latest/auto_examples/01_DataOperations/plot_download.html#sphx-glr-auto-examples-01-dataoperations-plot-download-py

Managing Data Projects

Data folders should have consistent structure (subjects, paradigm, metadata, etc)

Project folders can be more flexible with code to process data to final results

Consistency is critical

- Naming scheme
- Metadata

BIDS Standard for fMRI - http://bids.neuroimaging.io/, http://incf.github.io/bids-validator/

https://eglerean.wordpress.com/2017/05/24/project-management-data-management/

Principles of Coding

Don’t Repeat Yourself

Simplicity

Optimization

Don’t Repeat Yourself

Replicating code makes it harder to maintain and debug

Don’t cut and paste! Even though it seems faster in short run it will ultimately be more time-consuming in long run

Don’t reinvent the wheel. Use other people’s packages when possible (especially ones that have tests!)

Simplicity: the art of maximizing the amount of work not done

+ Always keep in mind the **reason** you are writing the code
+ Try to solve that problem only, with as little work as possible
+ Do not try to solve every conceivable problem vaguely related to your problem
+ Keep in mind the use cases you want to support
+ Do not try to support every conceivable use case
+ Detect issues as early in the code as possible (ideally before the code runs for a long time) and throw an error if something unexpected has happened

Optimization

“Premature optimization is the root of all evil” Donald Knuth

Profile your code

Start with the easy stuff

Parallelize repeated independent processes (e.g., resampling, different subjects)

Vectorize