Reproducible Research

bioconnector.org/workshops/lessons/r/r-repres

September 29, 2015

Who cares about reproducible research?

Science is plagued by reproducibility problems. Especially genomics!

- Scientists in the United States spend $28 billion each year on basic biomedical research that cannot be repeated successfully.[1]
- A reproducibility study in psychology found that only 39 of 100 studies could be reproduced.[2]
- The Journal *Nature* on the issue of reproducibility:[3]
  - "*Nature* and the Nature research journals will introduce editorial measures to address the problem by improving the consistency and quality of reporting in life-sciences articles… **we will give more space to methods sections. We will examine statistics more closely and encourage authors to be transparent, for example by including their raw data.**"
  - *Nature* also released a checklist, unfortunately with *wimpy* computational check *(see #18)*.
- On microarray reproducibility:[4]
  - 18 Nat. Genet. microarray experiments
  - Less than 50% reproducible
  - Problems:
    - Missing data (38%)
    - Missing software/hardware details (50%)
    - Missing method/processing details (66%)

[1] Freedman, et al. "The economics of reproducibility in preclinical research." *PLoS Biol* 13.6 (2015): e1002165.

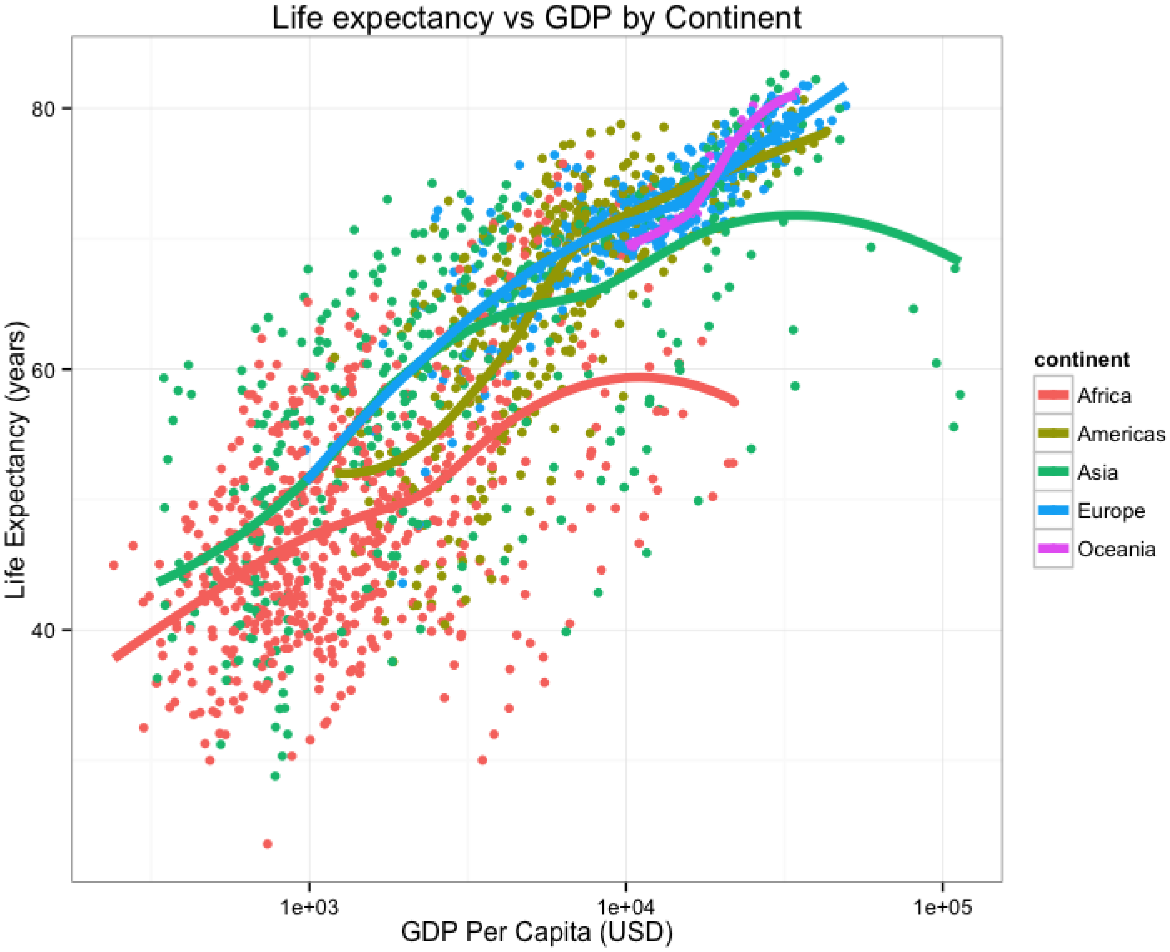
[2] http://www.nature.com/news/first-results-from-psychology-s-largest-reproducibility-test-1.17433

[3] http://www.nature.com/news/reproducibility-1.17552

[4] Ioannidis, John PA, et al. "Repeatability of published microarray gene expression analyses." *Nature genetics* 41.2 (2009): 149-155.

- NGS: run-of-the-mill variant calling (align, process, call variants):[5]
    - 299 articles published in 2011 citing the 1000 Genomes project pilot publication
    - Only 19 were NGS studies with similar design
    - Only 10 used tools recommended by 1000G.
    - Only 4 used full 1000G workflow (realignment & quality score recalibration).

Consider this figure:



How do we reproduce it? What do we need?

- The data
    - Data points themselves
    - Other metadata
- The code

[5] Nekrutenko, Anton, and James Taylor. "Next-generation sequencing data interpretation: enhancing reproducibility and accessibility." *Nature Reviews Genetics* 13.9 (2012): 667-672.

  - Should be readable
  - Comments in the code / well-documented so a normal person can figure out how it runs.
  - How were the trend lines drawn?
  - What version of software / packages were used?

This kind of information is rarely available in scientific publications, but it's now extraordinarly easy to put this kind of information on the web.

Could I replicate Figure 1 from your last publication? If not, what would *you and your co-authors* need to provide or do so I could replicate Figure 1 from your last publication?

As scientists we should aim for *robust* and *reproducible* research.

- "**Robust research** is about doing small things that stack the deck in your favor to prevent mistakes." *-Vince Buffalo, author of Bioinformatics Data Skills (2015).*
- **Reproducible research** can be repeated by other researchers with the same results.

Reproducibility is hard!

1. Genomics data is too large and high dimensional to easily inspect or visualize. Workflows involve multiple steps and it's hard to inspect every step.
2. Unlike in the wet lab, we don't always know what to expect of our genomics data analysis.
3. It can be hard to distinguish *good* from *bad* results.
4. Scientific code is usually only run once to generate results for a publication, and is more likely to contain silent bugs. (code that may produces unknowingly incorrect output rather than stopping with an error message).

What's in it for *you?*

Yeah, it takes a lot of effort to be robust and reproducible. However, *it will make your life (and science) easier!*

- Most likely, you will have to re-run your analysis more than once.

- In the future, you or a collaborator may have to re-visit part of the project.
- Your most likely collaborator is your future self, and your past self doesn't answer emails.
- You can make modularized parts of the project into re-useable tools for the future.

- Reproducibility makes you easier to work and collaborate with.

Some recommendations for reproducible research

1. **Write code for humans, write data for computers.**

   - Code should be broken down into small chunks that may be re-used.

   - Make names/variables consistent, distinctive and meaningful.

   - Adopt a style be consistent.[6]
   - Write concise and clear comments.

2. **Make incremental changes.** Work in small steps with frequent feedback. Use version control. See http://swcarpentry.github.io/git-novice/ for resources on version control.
3. **Make assertions and be loud, in code and in your methods.** Add tests in your code to make sure it's doing what you expect. See http://software-carpentry.org/v4/test/ for resources on testing code.
4. **Use existing libraries (packages) whenever possible.** Don't reinvent the wheel. Use functions that have already been developed and tested by others.
5. **Prevent catastrophe and help reproducibility by making your data *read-only.*** Rather than modifying your original data directly, always use a workflow that reads in data, processes/modifies, then writes out intermediate and final files as necessary.
6. **Encapsulate the full project into one directory that is supported with version control.** See: Noble, William Stafford. "A quick guide to organizing computational biology projects." *PLoS Comput Biol* 5.7 (2009): e1000424.
7. **Release your code and data.** Simple. Without your code and data, your research is not reproducible.

   - GitHub (https://github.com/) is a great place for storing, distributing, collaborating, and version-controlling code.
   - RPubs (http://rpubs.com/) allows you to share dynamic documents you write in RStudio online.
   - Figshare (http://figshare.com/) and Zenodo (https://zenodo.org/) allow you to upload any kind of research output, publishable or not, free and unlimited. Instantly get permanently available, citable DOI for your research output.
   - *"Data/code is available upon request"* or *"Data/code is available at the lab's website"* are completely unacceptable in the 21st century.

8. **Write code that uses relative paths.**

   - Don't use hard-coded absolute paths (i.e. `/Users/stephen/Data/seq-data.csv` or `C:\Stephen\Documents\Data\Project1\data.txt`).
   - Put the data in the project directory and reference it *relative* to where the code is, e.g., `data/gapminder.csv`, etc.

[6] http://adv-r.had.co.nz/Style.html

9. **Always set your seed.** If you're doing anything that involves random/monte-carlo approaches, always use `set.seed()`.
10. **Document everything and use code as documentation.**

    - Document why you do something, not mechanics.
    - Document your methods and workflows.
    - Document the origin of all data in your project directory.
    - Document **when** and **how** you downloaded the data.
    - Record **data** version info.
    - Record **software** version info with `session_info()`.
    - Use dynamic documentation to make your life easier.