REPRODUCIBLE RESEARCH

Matthew Flickinger, Ph.D.

CSG Tech Talk

July 14, 2016
Reproducible research:

The idea that data analyses, and more generally, scientific claims, are published with their data and software code so that others may verify the findings and build upon them.

From https://www.coursera.org/learn/reproducible-research
Reproducibility $\neq$ Replication
Reproducibility ≠ Replication

My code and data support the claims I make in my paper

I've independently replicated your results with a different data set
Not a new idea...

- Jon Claerbout in 1990's set out to make "reproducible documents."
- Claerbout believed that "an article about computational result is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result." (Buckheit and Donoho, 1995)
- Donald Knuth encouraged "literate programming" in the 1980's where code is mixed with prose that describes its intent
- AJ Rossini extended those ideas into "literate statistical practice" with R in particular (2001)
- Computational science combined tools from software development with traditional scientific analysis
Scientific gains from reproducibility

- Standard to judge scientific claims
  - Allows scrutiny
  - The code describes exactly what was done
- Avoid effort duplication and encourage cumulative knowledge development

Personal gains from reproducibility

• Better work habits (organization)
• Better teamwork (collaboration)
• Changes are easier (reactive)
• Higher research impact (more citations)

Reproducibility in scholarly publications

- Science published a special issue on the topic in Dec 2011
- Journal of Biostatistics has an associate editor for reproducibility
- Some journals only require a sufficient written description of code which can be used to recreate it
- Material and Methods sections are often far too short to provide all necessary critical details of a particular implementation
- Many journals still have no clear/explicit guidelines

Reproducible research (Titus Brown 2012)

Publication

Code and Data

Source Code

Reproducible Figures

Instructions

Data
Reproducible journalism (FiveThirtyEight)
Spectrum of reproducibility

Tools for reproducibility

• Code and documentation (literate programming)
  • Knitr (rmarkdown, pandoc, Sweave)
  • Jupyter Notebook (iPython Notebook, rNotebook)

• Version control and code sharing
  • git (SVN, mercurial)
  • github.com (bitbucket.com)

• Workflow coordination and dependency management
  • make (gnumake)
Literate programming

• Descriptions of code in plain English interspersed with actual code
• These files support two actions
  • "Tangle" – Extract executable code (machine readable)
  • "Weave" – Combine into document (human readable)
• Organize code into small, understandable sections
• Include pictures or figures to describe what's going on
Literate R programming with knitr

Include Text

We can now plot these values as a bar chart, adding a line that helps us identify deltas of more than a quarter of a second between cars.

Include R Code

```r
head(ddx,n=3)
```

```r
## driverNum time laps year natGap gap race pos driverName
## 1 4 98.021 19 2012 0.000 0.000 MALAYSIA 1 Lewis Hamilton
## 2 1 98.535 21 2012 0.514 0.514 MALAYSIA 2 Sebastian Vettel
## 3 8 98.813 21 2012 0.792 0.792 MALAYSIA 3 Nico Hülkenberg
```

Results

```r
g+ggplot(ddx) + geom_bar(aes(x=pos,y=delta),stat='identity')+geom_hline(yintercept=0.25,col='grey')
g+ggplot(ddx) + geom_bar(aes(x=pos,y=delta),stat='identity')+geom_hline(yintercept=0.25,col='grey')
```

Plots

```r
g+xlab('Position')+ylab('Lap delta from car ranked one position ahead (s)')
```

Returning to the laps count, how might we more correctly render the corresponding y-axis value. In the following.

```r
(r sortedLapcountbarChart,fig.cap='Lap delta bar chart')
```
Knitr good and bad

• Pros
  • Easy integration with Rstudio
  • Works with plain text files
  • Great for reproducible reports
  • Can automate with knit2*() functions in R code

• Not ideal for
  • Long running computations
  • Very precise formatting
Literate programming with Jupyter Notebook
Jupyter Notebooks

- Notebooks are stored in plain text as JSON documents
- More interactive HTML interface (use in web browser)
- Support for different computation "kernels"
  - Python
  - R
- See reference from Ryan's presentation on the Tech Talk wiki
Version control

• Track changes to your files and scripts over time
• Include messages about why changes were made
• Easily return to old versions of files
Version control with git

• The program "git" has become widespread for version control
• Built-in support for git is included in Rstudio
• Command-line tool for tracking how files change
• Many graphical user interfaces (GUIs) available to make working with git easier
git best practices

- git allows you to "commit" changes to your files
- Each commit is accompanied by a git message
- Make small changes at a time; include a descriptive change message
  - Helpful to understand why things change (stored in the log)
  - Bad: "fixed stuff"
  - Good: "Add MAF filter to SNPs"
- Use branches to "try stuff out"
  - Test out changes to a file or analysis variations
  - Easily switch between branches
Share code on github.com

- You can publish your git project to github.com (or bitbucket.com, etc)
- Other can see your code and
  - Send fixes
  - Report issues
  - "Fork" and use with their data
- You will have a backup of your work "in the cloud"
- Public sharing is free, private repositories cost money
Execution automation

• What if you need to run several different programs for your analysis
• How do you let others know the order that things need to be run
• Writing a script file is good, but it will always run all tasks in the file
Using "make" to perform an analysis

• The program make (or gnumake) was created to manage the compiling of source code into an executable program

• make files contain "recipes" to build "target" files based on a list of "prerequisites"

• make looks at the timestamps of the files involved and will rebuild targets if they are older than any of the prerequisites
Sample make file

R_OPTS=--vanilla

mypaper.pdf: mypaper.bib mypaper.tex Figs/fig1.pdf Figs/fig2.pdf
  pdflatex mypaper
  bibtex mypaper
  pdflatex mypaper
  pdflatex mypaper

Figs/%.pdf: R/%.R
  Rscript $(R_OPTS) $< $@

Variables

Target

Prerequisites

Recipe

Wild Card Patterns

Variable: First Prereq

Variable: Target
Data archiving

• How do I share my data?
• Your data may be too large to share on github.com
• Unstructured repositories
  • Figshare (https://figshare.com/) 20GB free private space, unlimited public space, max file size 5GB
  • Dryad (http://datadryad.org/) $120 upon publication up to 20GB
• Specialty repositories
  • Genbank, NCBI Read Archive, dbSNP, dbVar, Gene Expression Omnibus, etc
Reproducibility isn't easy

...but neither is writing a good paper

You get better with practice
Reproducible anywhere?

- It can be very difficult to cleanly pack up your code so it runs on any other computer
- Software versions change over time
- Big differences between operating systems
- Not uncommon for published software to fail to repeat (even in the CS field, see image)
- R package "packrat" can help with dependency management

http://reproducibility.cs.arizona.edu/
Reproducibility can save you time

• Change is inevitable
• Ask yourself
  • If I need to drop 10 samples, how quickly can I recreate my figures?
  • How quickly can I run this same analysis in a different data set
• If you think about reproducibility from the beginning, these tasks should be easy
Reproducible from the start

• Reproducibility requires forethought
• Much more difficult to "add reproducibility" at the end of an analysis
• Enables easier hypothesis testing during your own analysis
Automate everything

• Make sure there is a script or make recipe for every file you create
  • Where did this file come from?
  • Why does it have 5 fewer samples than my file?
• Track all data files and available meta data
• Avoid steps you can’t automate
  • If you need to point-and-click on something, it's difficult to automate
  • Move to the beginning or end of your pipeline
• Use seeds when you need random numbers
Further Resources for Learning

- Reproducible Research
  - Coursera (R, free to audit course) ([https://www.coursera.org/learn/reproducible-research](https://www.coursera.org/learn/reproducible-research))
- Git
  - Try Git ([https://try.github.io/](https://try.github.io/))
- Make
  - Minimal make ([http://kbroman.org/minimal_make/](http://kbroman.org/minimal_make/))
  - Reproducible bioinformatics pipelines using make ([https://bsmith89.github.io/make-bml/](https://bsmith89.github.io/make-bml/))
THANK YOU