



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.

Reproducible



Reliable

Robust

Reusable

Reproducibility



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 associated 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 guidelines¹



1) Stodden, Victoria, Peixuan Guo, and Zhaokun Ma. "Toward reproducible computational research: an empirical analysis of data and code policy adoption by journals." *PloS one* 8.6 (2013): e67111.

Reproducible research (Titus Brown 2012)

Publication

1203.4802v2.pdf

arxiv.org/pdf/1203.4802v2.pdf

1203.4802v2.pdf 1 / 18

A Reference-Free Algorithm for Computational Normalization of Shotgun Sequencing Data

C. Titus Brown^{1,2,*}, Adina Howe², Qingpeng Zhang¹, Alexis B. Pyrkosz², Timothy H. Brom¹

1 Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
2 Microbiology and Molecular Genetics, Michigan State University, East Lansing, MI, USA
3 USDA Avian Disease and Oncology Laboratory, East Lansing, MI, USA
* E-mail: ctb@msu.edu

Abstract

Deep shotgun sequencing and analysis of genomes, transcriptomes, amplified single-cell genomes, and metagenomes has enabled investigation of a wide range of organisms and ecosystems. However, sampling variation in short-read data sets and high sequencing error rates of modern sequencers present many new computational challenges in data interpretation. These challenges have led to the development of new classes of mapping tools and *de novo* assemblers. These algorithms are challenged by the continued improvement in sequencing throughput. We here describe digital normalization, a single-pass computational algorithm that systematizes coverage in shotgun sequencing data sets, thereby decreasing sampling variation, discarding redundant data, and removing the majority of errors. Digital normalization substantially reduces the size of shotgun data sets and decreases the memory and time requirements for *de novo* sequence assembly, all without significantly impacting content of the generated contigs. We apply digital normalization to the assembly of microbial genomic data, amplified single-cell genomic data, and transcriptomic data. Our implementation is freely available for use and modification.

Author Summary

Introduction

The ongoing improvements in DNA sequencing technologies have led to a new problem: how do we analyze the resulting large sequence data sets quickly and efficiently? These data sets contain millions to billions of short reads with high error rates and substantial sampling biases [1]. The vast quantities of deep sequencing data produced by these new sequencing technologies are driving computational biology to extend and adapt previous approaches to sequence analysis. In particular, the widespread use of deep shotgun sequencing on previously unsequenced genomes, transcriptomes, and metagenomes, has resulted in the development of several new approaches to *de novo* sequence assembly [2].

There are two basic challenges in analyzing short-read sequences from shotgun sequencing. First, deep sequencing is needed for complete sampling. This is because shotgun sequencing samples randomly from a population of molecules; this sampling is biased by sample content and sample preparation, requiring even deeper sequencing. A human genome may require 100x coverage or more for near-complete sampling, leading to shotgun data sets 300 GB or larger in size [3]. Since the lowest abundance molecule determines the depth of coverage required for complete sampling, transcriptomes and metagenomes containing rare species also present challenges for complete sampling.

arXiv:1203.4802v2 [q-bio.GN] 21 May 2012

Code and Data

1203.4802v2.pdf

ged.msu.edu/papers/2012-diginorm/

Title: A Reference-Free Algorithm for Computational Normalization of Shotgun Sequencing Data

C. Titus Brown, Adina Howe, Qingpeng Zhang, Alexis B. Pyrkosz, and Timothy H. Brom

arXiv preprint

Deep shotgun sequencing and analysis of genomes, transcriptomes, amplified single-cell genomes, and metagenomes has enabled investigation of a wide range of organisms and ecosystems. However, sampling variation in short-read data sets and high sequencing error rates of modern sequencers present many new computational challenges in data interpretation.

Online resources and data:

- **A tutorial for running khmer on microbial genomes and eukaryotic transcriptomes.**
- **Git repository for khmer:** github.com/ged-lab/khmer/tree/master/diginorm
- **Git repository for paper & data analysis pipeline:** github.com/ged-lab/2012-paper-diginorm
- **Instructions on running the paper analysis pipeline & reproducing the figures in the paper**
- **HTML view of the ipython notebook containing code and scripts to reproduce the figures in the paper. (See the pipeline notes for a runnable version.)**
- **Data required to run the pipeline (.tar.gz, 7.9gb)**
- **Assembled microbial genomes and eukaryotic transcriptomes (.tar.gz, 110 mb)**

Source Code

Reproducible Figures

Instructions

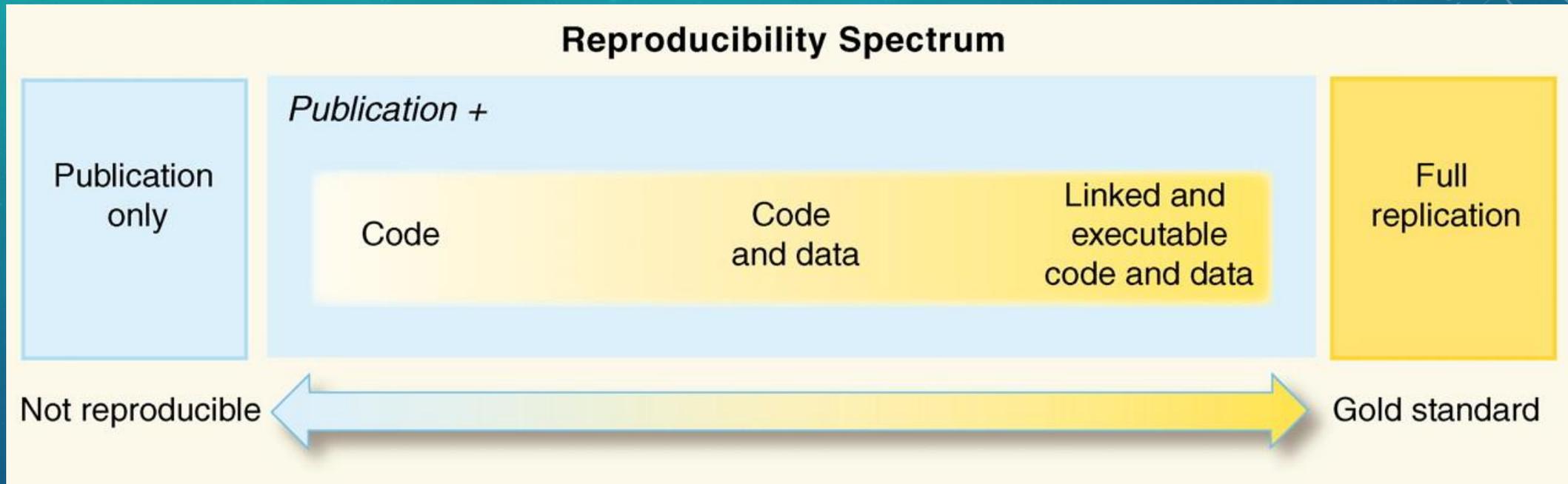
Data

Reproducible journalism (FiveThirtyEight)

The screenshot shows the FiveThirtyEight website interface. At the top, there's a navigation bar with categories like Politics, Sports, Science & Health, Economics, and Culture. Below the navigation is a large illustration featuring four name cards: one blank, one with 'MARIA GARCIA', and one with 'JAMES SMITH'. The article title is 'Dear Mona, What's The Most Common Name In America?' by Mona Chalabi and Andrew Flowers. The article text begins with 'Dear Mona, What are the most common first- and last-name combinations in the United States? Is John Smith really the most common name?'. A 'RECOMMENDED' sidebar on the right lists other articles like 'The Loudest Sound In The World Would Kill You On The Spot' and 'Bad News, Cavs Fans: The Warriors Don't Care How Good Your Team Is'.

The screenshot shows a GitHub repository page for 'fivethirtyeight / data'. The repository contains a list of files and folders, including 'README.md', 'adjusted-name-combinations-list.csv', 'adjusted-name-combinations-matrix.csv', 'adjustments.csv', 'aging-curve.csv', 'independent-name-combinations-by-pop.csv', 'most-common-name.R', 'new-top-firstNames.csv', 'new-top-surnames.csv', 'state-pop.csv', and 'surnames.csv'. Each file has a description, such as 'most common name data and script'. The repository is on the 'master' branch and has 8 issues, 2 pull requests, and 635 watchers.

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

```
152  
153 We can now plot these values as a bar chart, adding  
    quarter of a second, for example, between cars.
```

```
154  
155 - ```{r lapdeltas,fig.cap='Laptime deltas between cars'}  
156 g=ggplot(ddx) + geom_bar(aes(x=pos,y=delta),stat='identity')  
157 g+xlabs('Position')+ylabs('Lap delta from car ranked one position ahead (s)')  
158 ```
```

Include R Code

```
161  
162 - ```{r lapdeltasColoured, fig.cap='Lap delta bar chart'}  
163 ggplot(ddx) + geom_hline(yintercept=0.25,col='grey') +  
164   geom_bar(aes(x=pos,y=delta,col=factor(delta<0.25)),stat='identity')  
165 - ```
```

```
166 Returning to the laps count, how might we more correctly  
    being correctly rendered as a number, and we also need  
    basis of the corresponding y-axis value. In the following  
    count.
```

```
167  
168 - ```{r sortedLapcountbarChart,fig.cap='Lap count bar chart'}  
169 dd$laps = as.integer(dd$laps)  
170 g = ggplot(dd)+geom_bar(aes(x=reorder(driverName,-laps),y=laps),  
    element_text(angle = 45, hjust = 1))
```

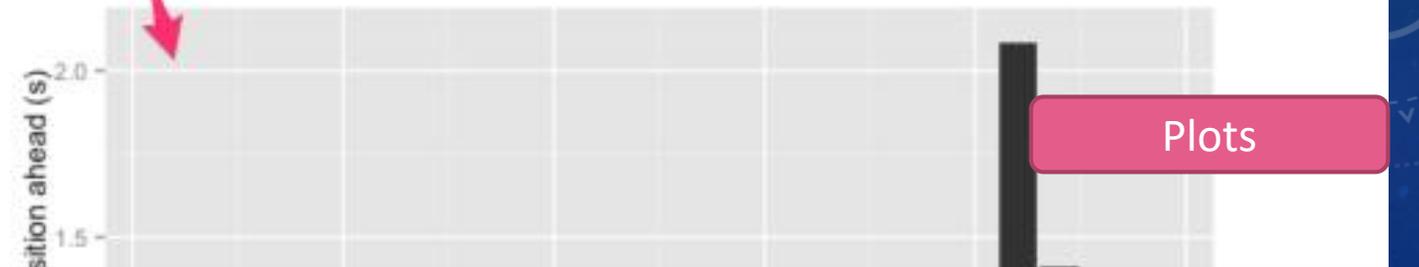
```
head(ddx,n=3)
```

```
## 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 Rosberg  
##  
##          team natTime percent delta  
## 1 McLaren-Mercedes 1:38.021 100.0000 0.000  
## 2 Red Bull Racing-Renault 1:38.535 100.5244 0.514  
## 3 Mercedes 1:38.813 100.8080 0.278
```

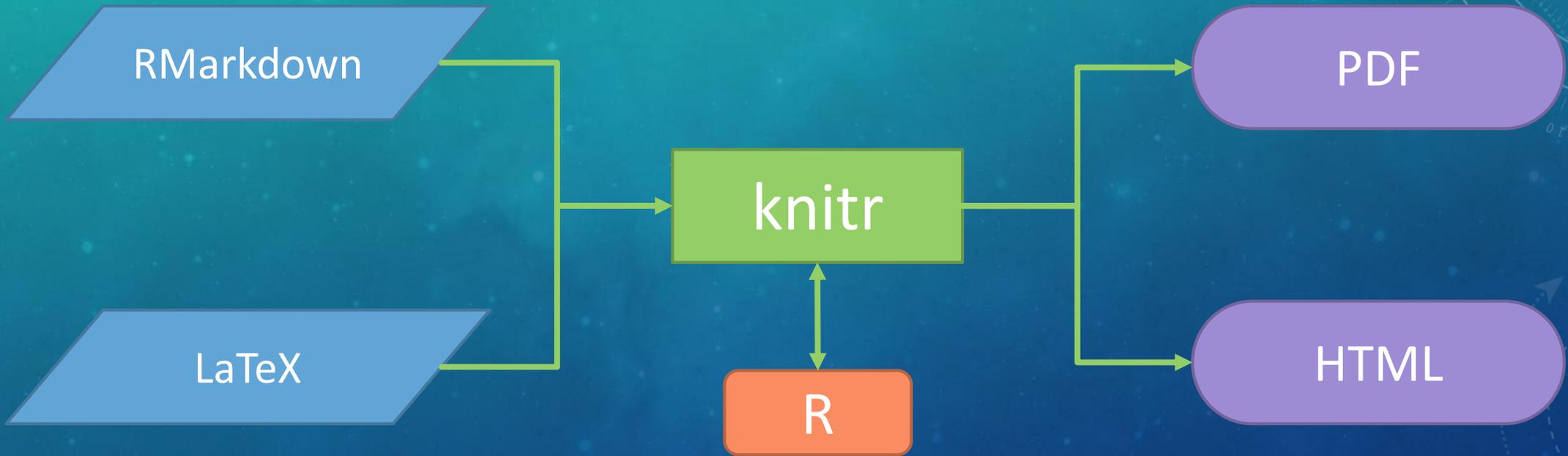
Results

```
161 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.
```

```
g=ggplot(ddx) + geom_bar(aes(x=pos,y=delta),stat='identity') + geom_hline(yintercept=0.25,col='grey')  
g+xlabs('Position')+ylabs('Lap delta from car ranked one position ahead (s)')
```



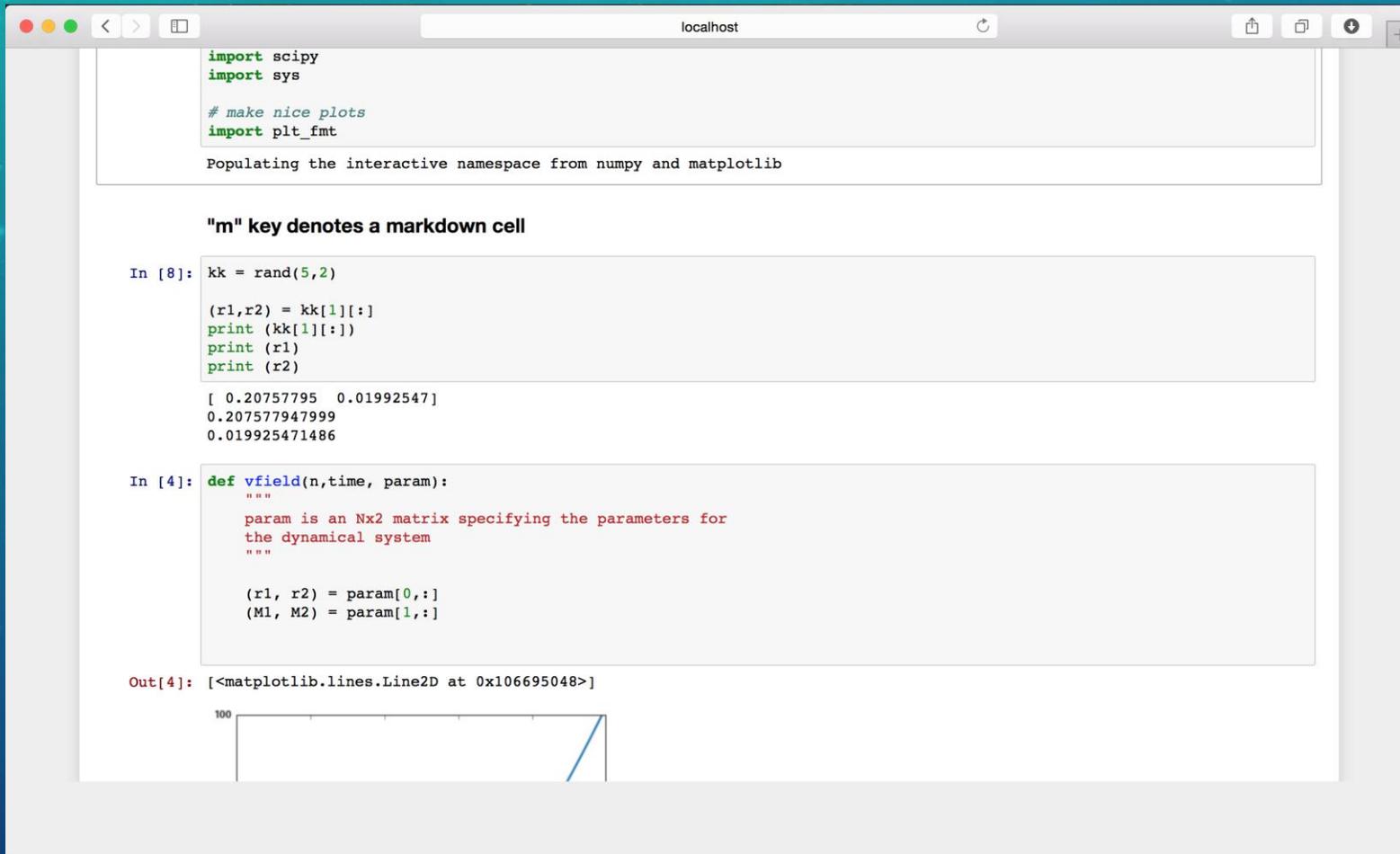
Plots



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



The screenshot shows a Jupyter Notebook window with the following content:

```
import scipy
import sys

# make nice plots
import plt_fmt

Populating the interactive namespace from numpy and matplotlib
```

"m" key denotes a markdown cell

In [8]: `kk = rand(5,2)`

```
(r1,r2) = kk[1][:]
print (kk[1][:])
print (r1)
print (r2)
```

```
[ 0.20757795  0.01992547]
0.207577947999
0.019925471486
```

In [4]: `def vfield(n,time, param):`

```
"""
    param is an Nx2 matrix specifying the parameters for
    the dynamical system
    """

    (r1, r2) = param[0,:]
    (M1, M2) = param[1,:]
```

Out[4]: [`matplotlib.lines.Line2D` at `0x106695048`]

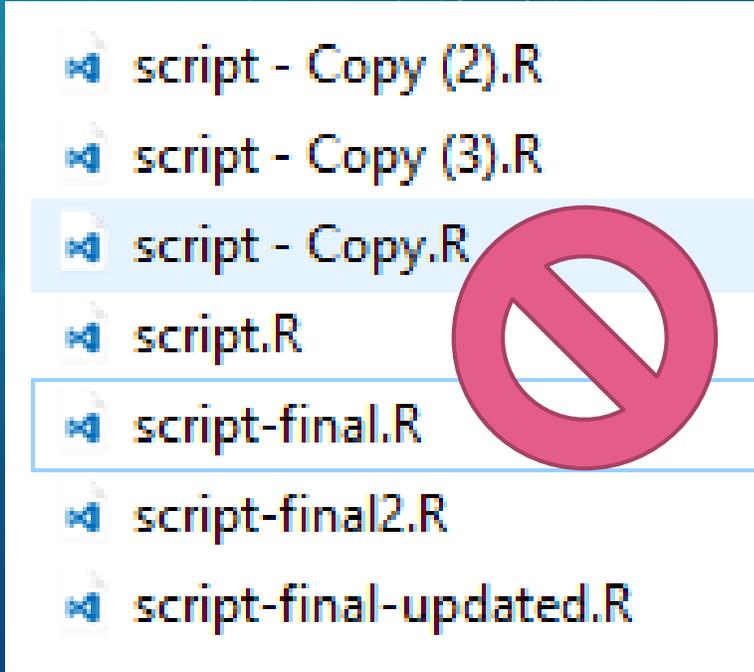


Jupyter Notebooks

- 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



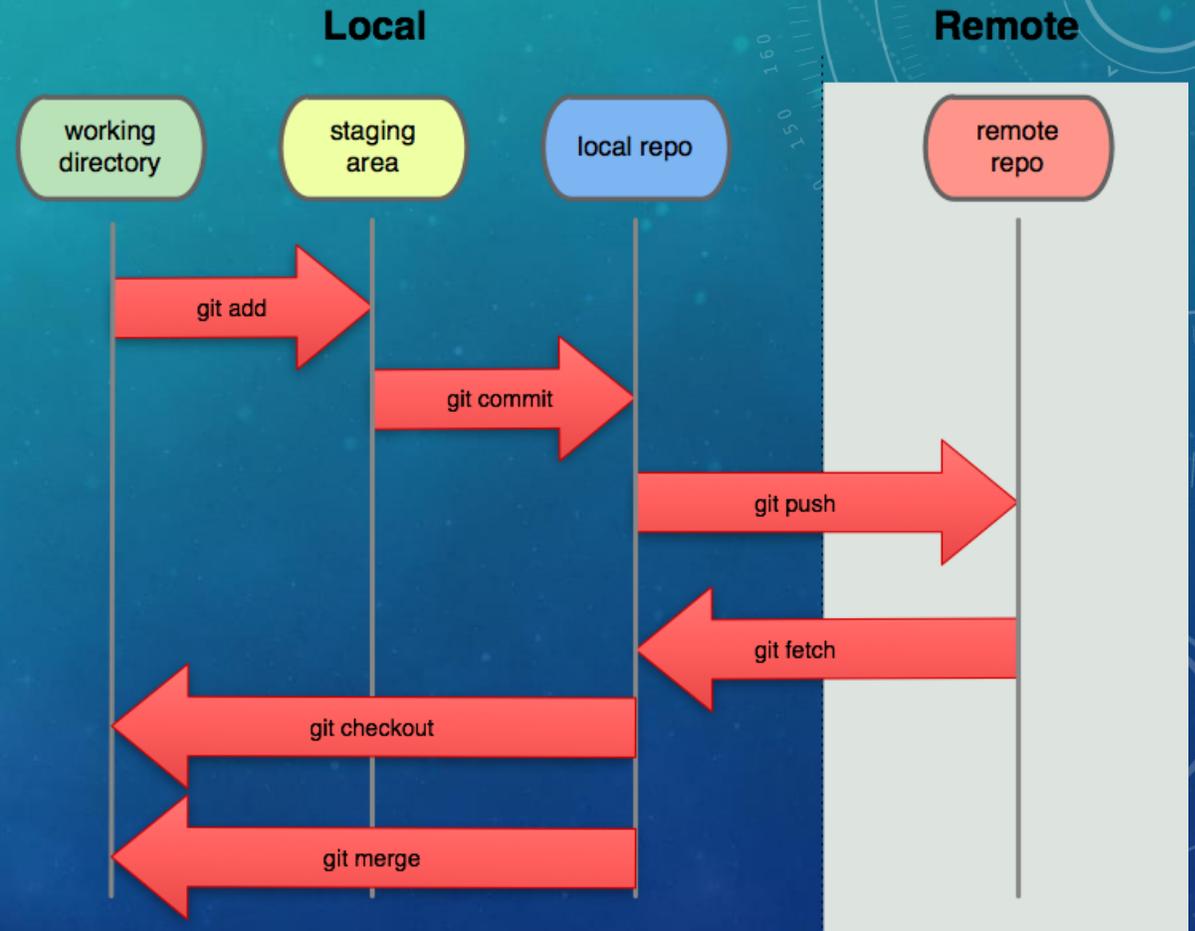
A screenshot of a file explorer window showing a list of files. The files are:

- script - Copy (2).R
- script - Copy (3).R
- script - Copy.R
- script.R
- script-final.R
- script-final2.R
- script-final-updated.R

The file 'script - Copy.R' is highlighted with a blue selection bar. A large red prohibition sign (a circle with a diagonal slash) is overlaid on the right side of the list, specifically covering the 'script - Copy.R' and 'script.R' entries.

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

```
* commit 9c8e9fd335381fe6a97708f7b3cd1d5acf670d2d
| \ Merge: 8aba87e... 6041ddd...
| | Author: Nick Quaranto <nick@quaran.to>
| | Date: Sun Jan 25 13:22:03 2009 -0500
| |
| | Fixing conflict!
| |
| * commit 6041dddac354fff0feec911e75a575082d8addb8
| | Author: Nick Quaranto <nick@quaran.to>
| | Date: Sun Jan 25 13:10:23 2009 -0500
| |
| | Changing cutoff default
| |
| * commit 8aba87e2e24744b7d1941e104b35033b9e2dbab5
| / Author: Nick Quaranto <nick@quaran.to>
| | Date: Sun Jan 25 13:16:04 2009 -0500
| |
| | Causing a merge on purpose
| |
| * commit 670e3538533554d0643ca128428997c98eb5d54e
| | Author: Nick Quaranto <nick@quaran.to>
| | Date: Sun Jan 25 13:04:30 2009 -0500
| |
| | Adding cutoff method to string
```

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 "recipies" 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

Variables

```
R_OPTS=--vanilla
```

Prerequisites

```
mypaper.pdf: mypaper.bib mypaper.tex Figs/fig1.pdf Figs/fig2.pdf
```

Target

```
pdflatex mypaper  
bibtex mypaper  
pdflatex mypaper  
pdflatex mypaper
```

Recipe

Wild Card Patterns

```
Figs/%.pdf: R/%.R
```

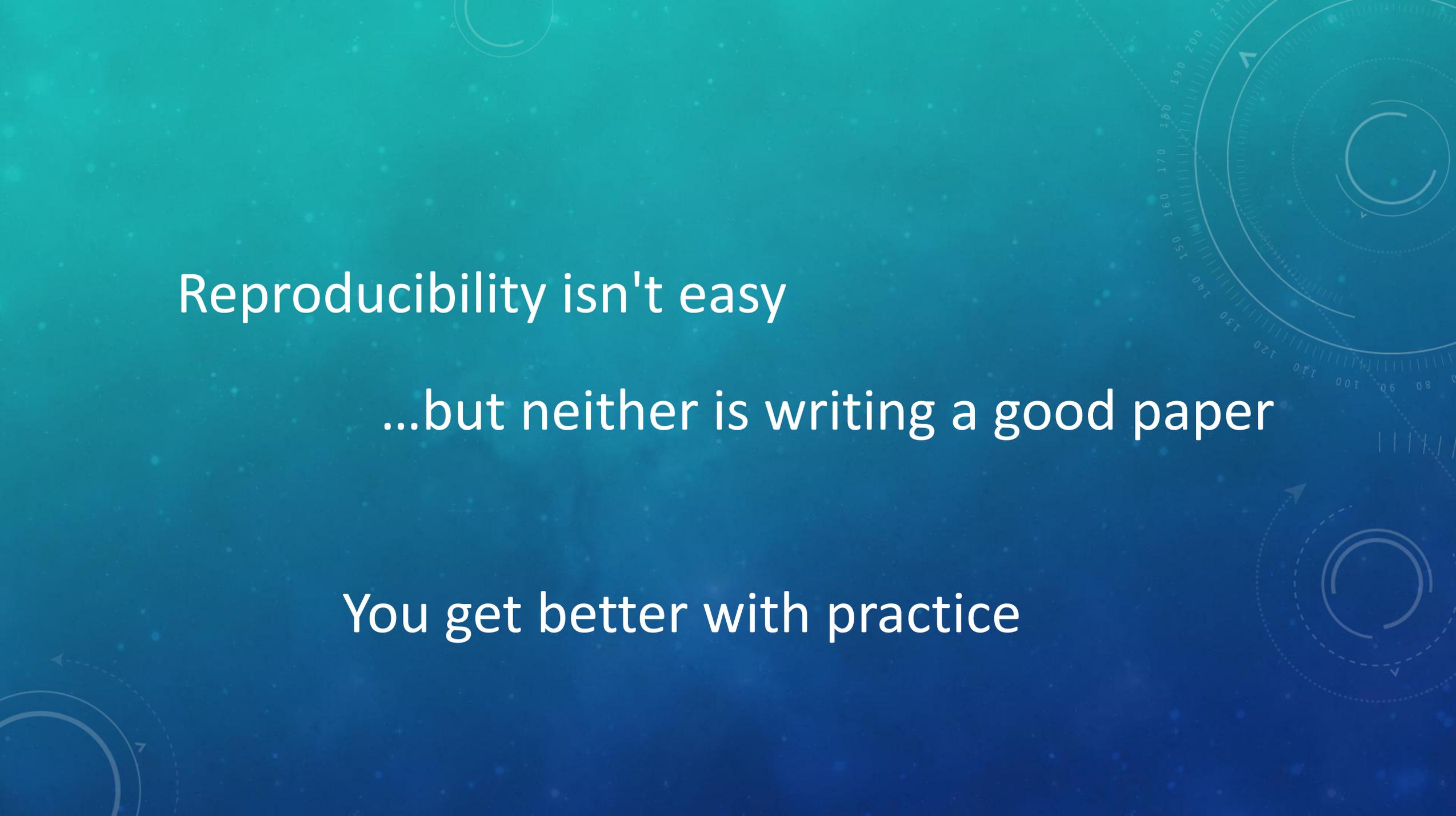
Variable: First Preref

```
Rscript $(R_OPTS) $< $@
```

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

The background is a teal-to-blue gradient with faint circular patterns and a scale on the right side. The scale has numbers from 80 to 200. There are also some dashed lines and arrows in the background.

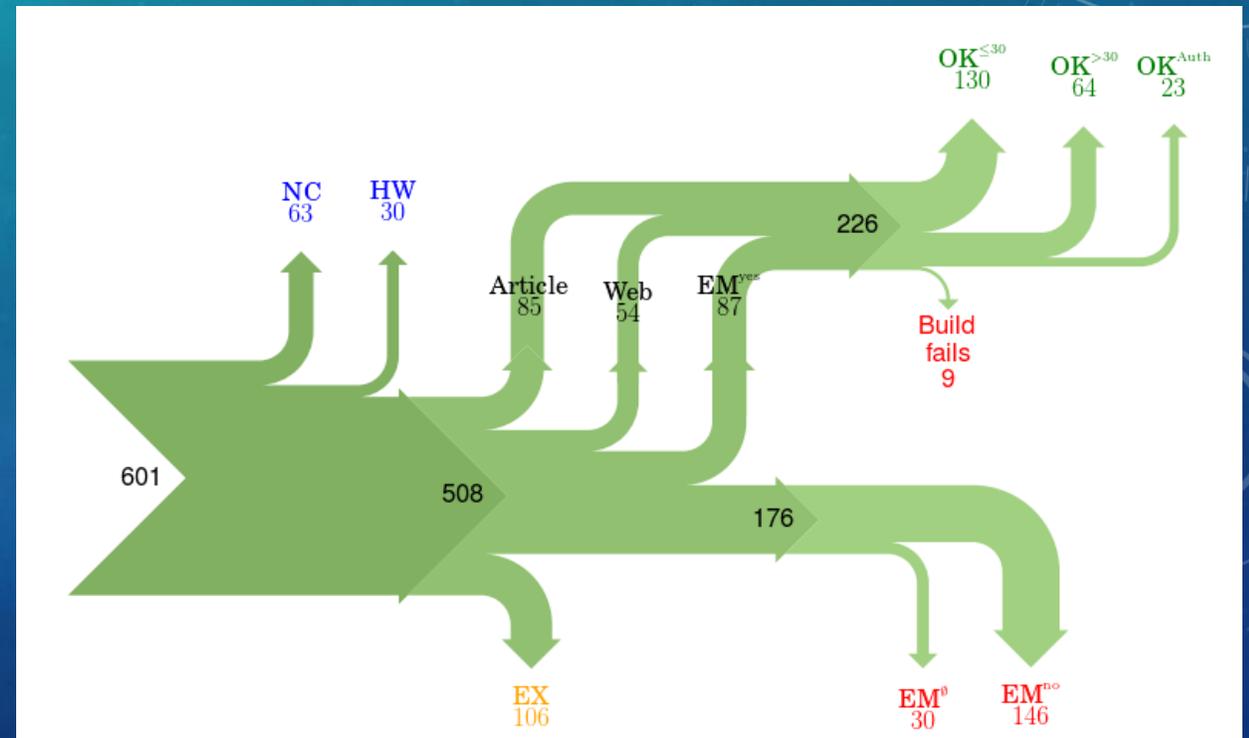
Reproducibility isn't easy

...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
- R package "packrat" can help with dependency management



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 in R
 - Coursera (free to audit course) (<https://www.coursera.org/learn/reproducible-research>)
 - Tools for Reproducible Research (<http://kbroman.org/Tools4RR/>)
- Git
 - Pro Git Book (<https://git-scm.com/book/>)
 - Try Git (<https://try.github.io/>)
- Make
 - Minimal make (http://kbroman.org/minimal_make/)
 - Reproducible bioinformatics pipelines using make (<https://bsmith89.github.io/make-bml/>)



THANK YOU