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

Reproducible

Reliable

Robust

Reusable

Reproducibility  Replication

Reproducibility

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

Replication

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

Gandrud, Christopher. Reproducible Research with R and R Studio. CRC Press, 2013.

Personal gains from reproducibility

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

Gandrud, Christopher. Reproducible Research with R and R Studio. CRC Press, 2013.

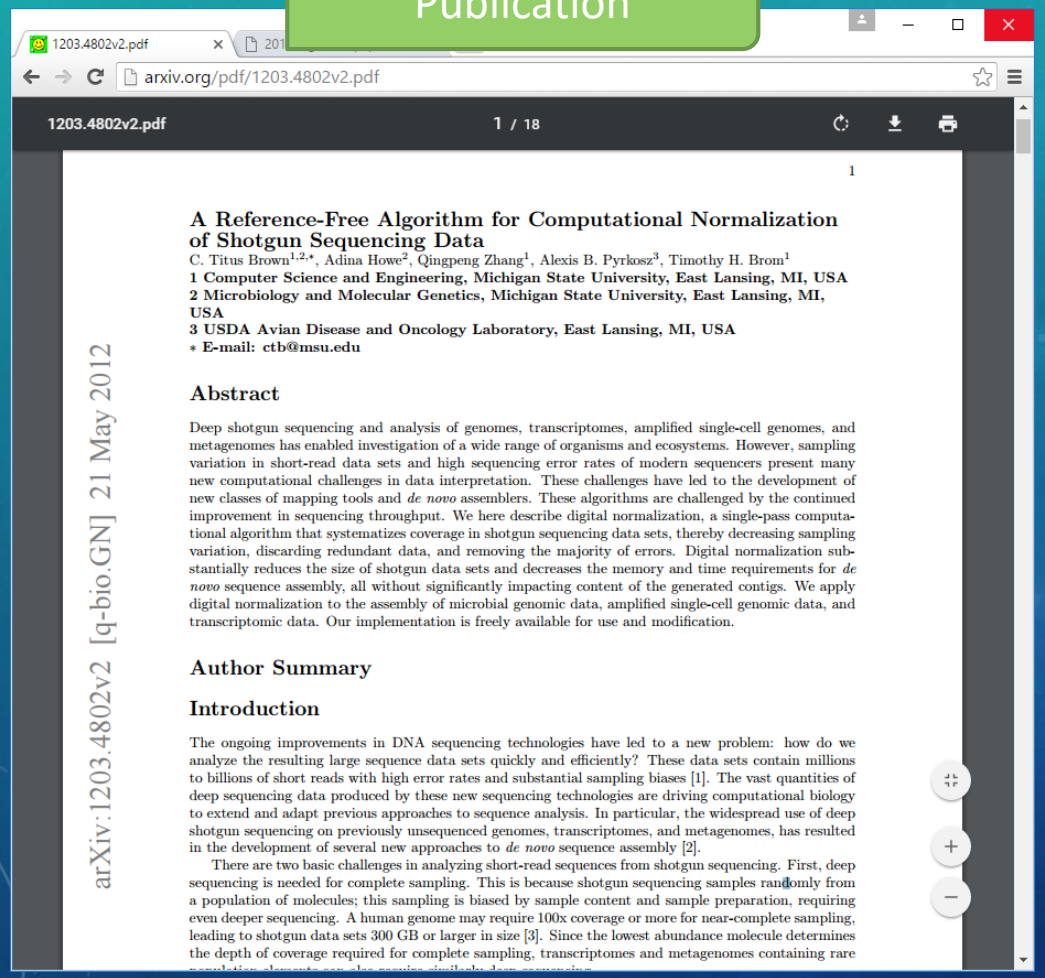
Reproducibility in scholarly publications

- 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 guidelines[1]

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

1

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

C. Titus Brown[1,2,*], Adina Howe[2], Qingpeng Zhang[1], Alexis B. Pyrkosz[3], Timothy H. Brom[1]
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.

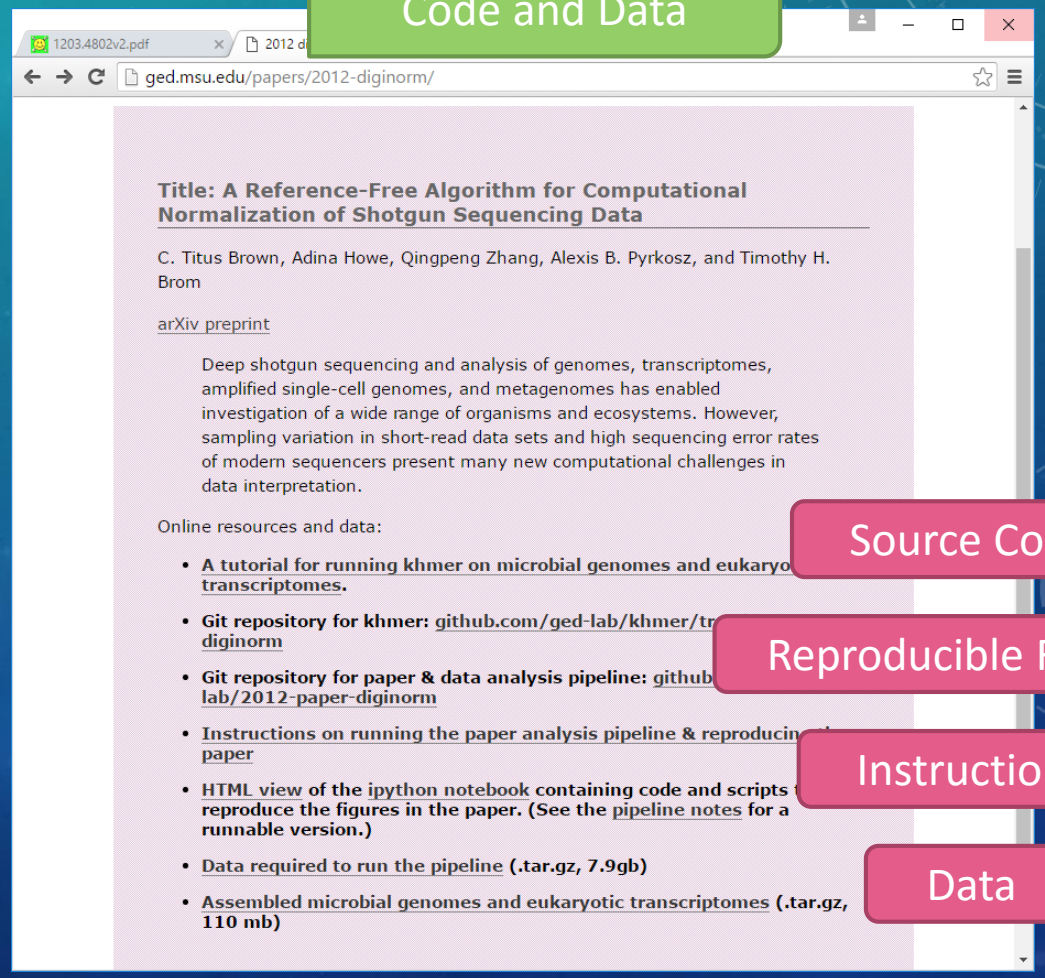
Author Summary

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

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 eukaryo transcriptomes.**
- **Git repository for khmer:** github.com/ged-lab/khmer/tr diginorm
- **Git repository for paper & data analysis pipeline:** github lab/2012-paper-diginorm
- **Instructions on running the paper analysis pipeline & reproducin paper**
- **HTML view of the ipython notebook containing code and scripts 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)**

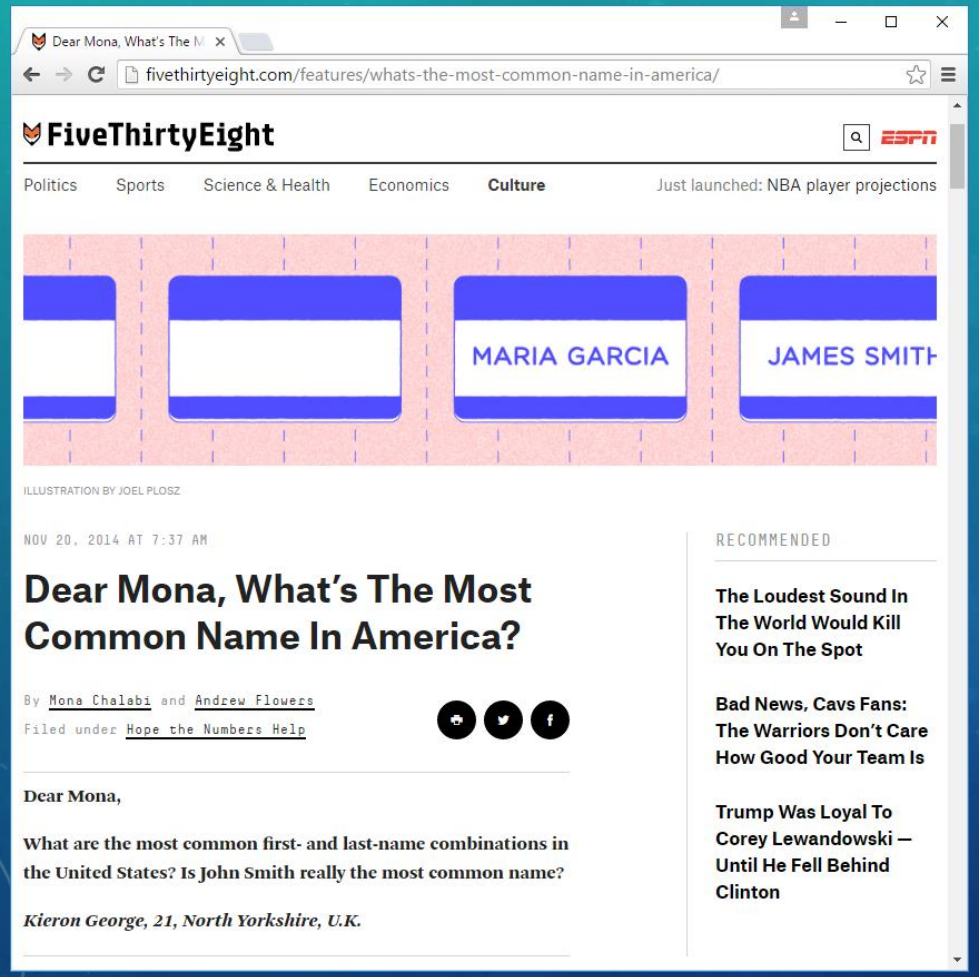
Source Code

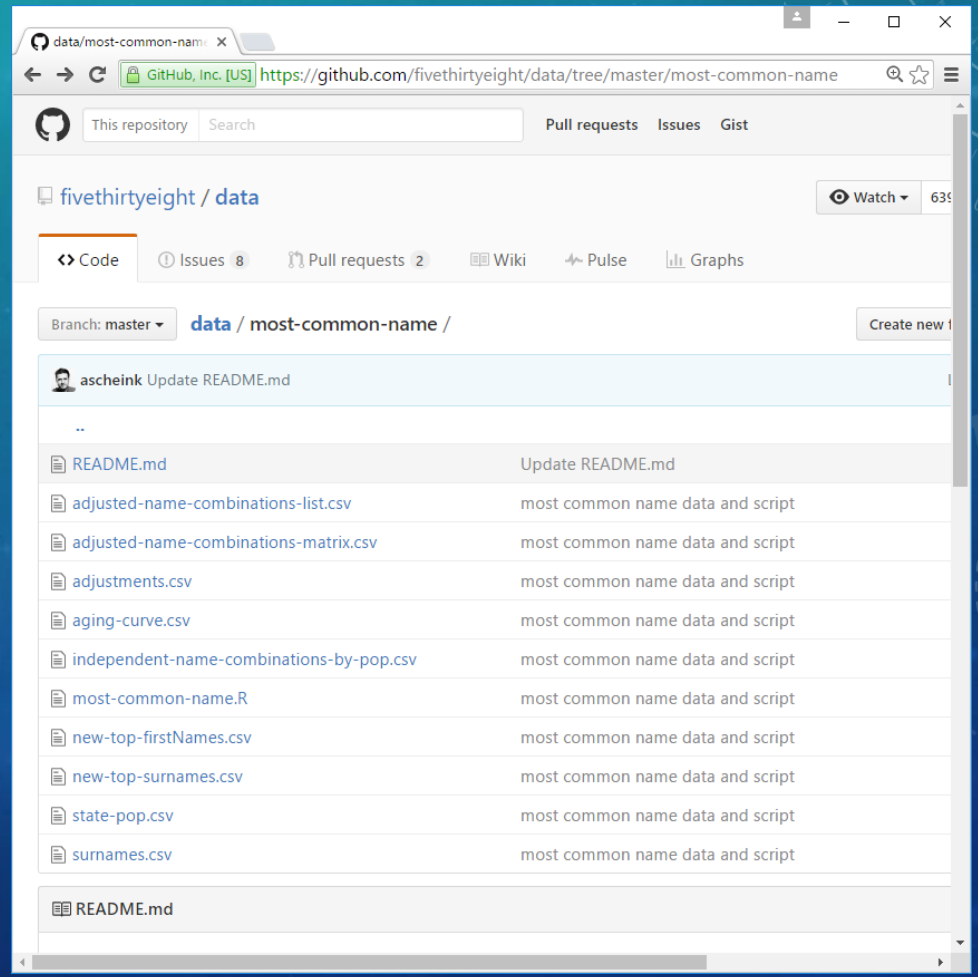
Reproducible Figures

Instructions

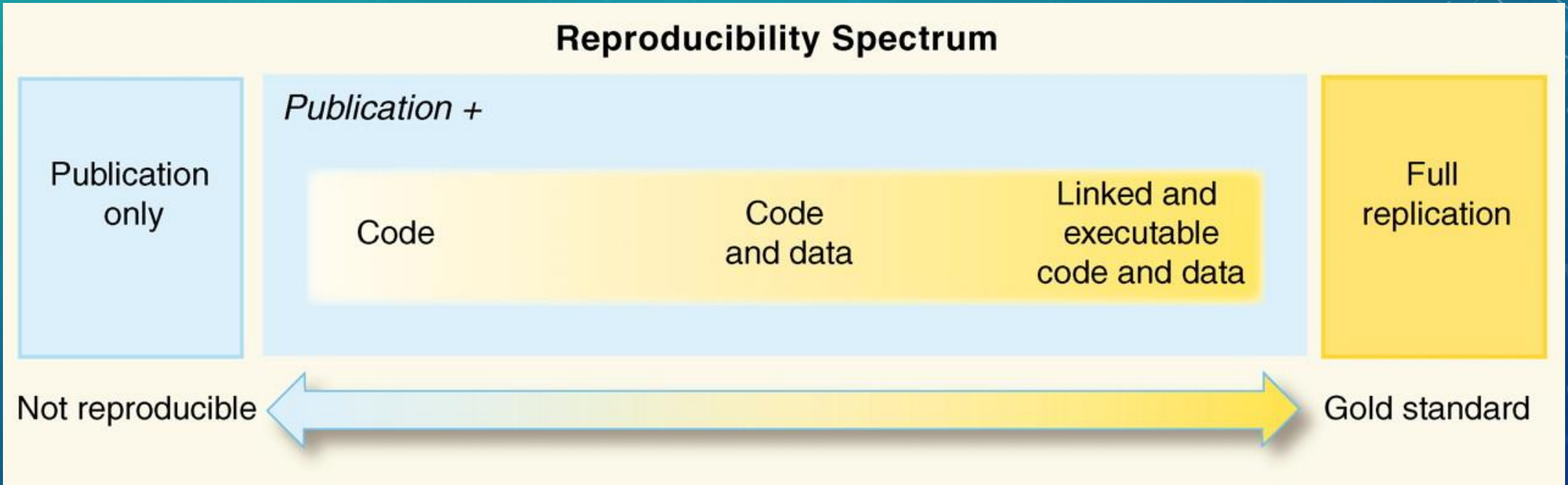
Data

Reproducible journalism (FiveThirtyEight)





Spectrum of reproducibility



Peng, Roger D. "Reproducible research in computational science." *Science* 334.6060 (2011): 1226-1227.

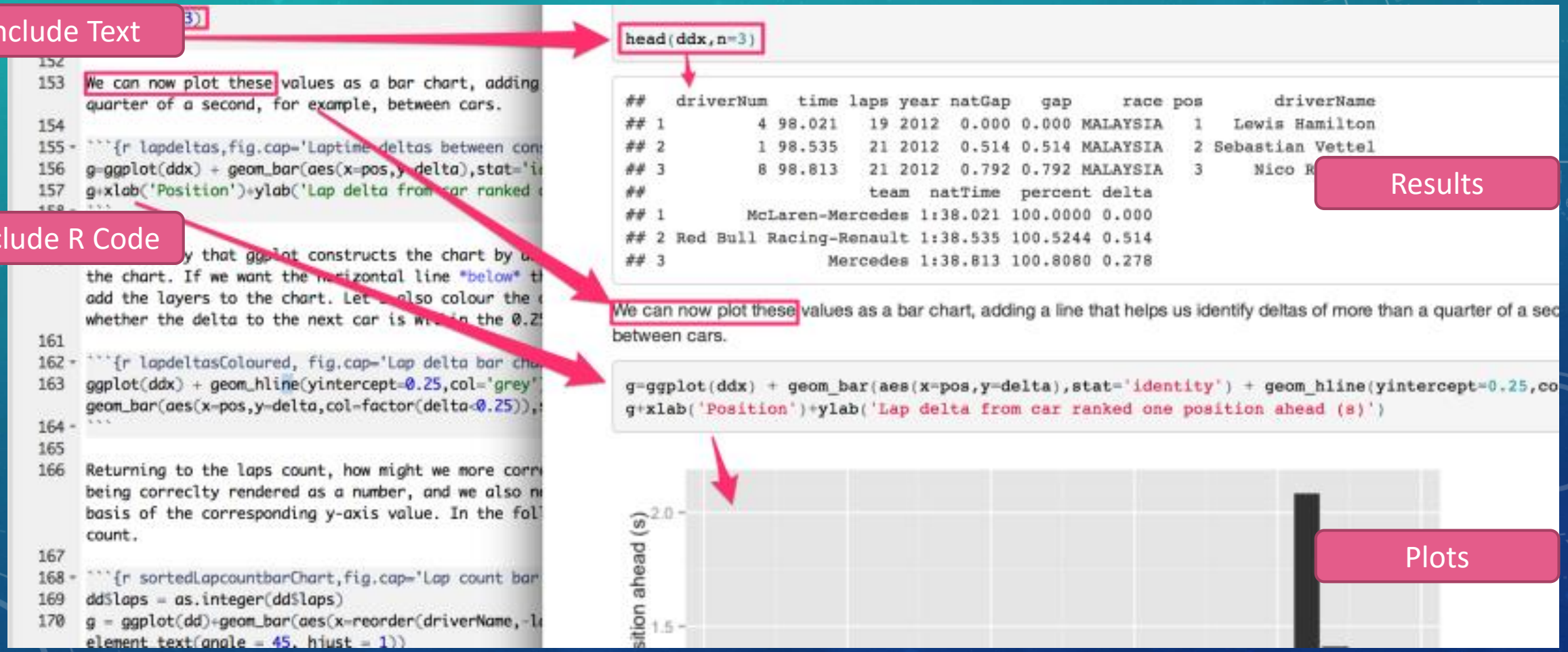
Tools for reproducibility

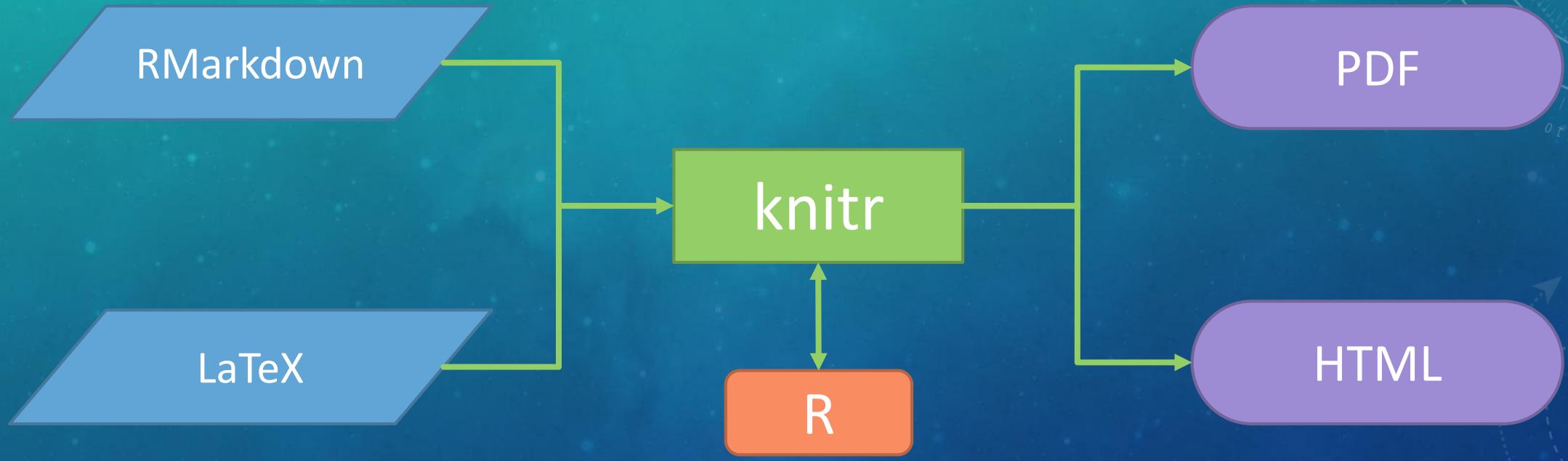
- Code and documentation (literate programming)
  - Knitr (rmarkdown, pandoc, Sweave)
  - Jupyter Notebook (iPython Notebook, rNotebook)
- Version control and code sharing
  - git (SVN, mercurial)
  - github.com (bitbucket.com)
- Workflow coordination and dependency management
  - make (gnumake)

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

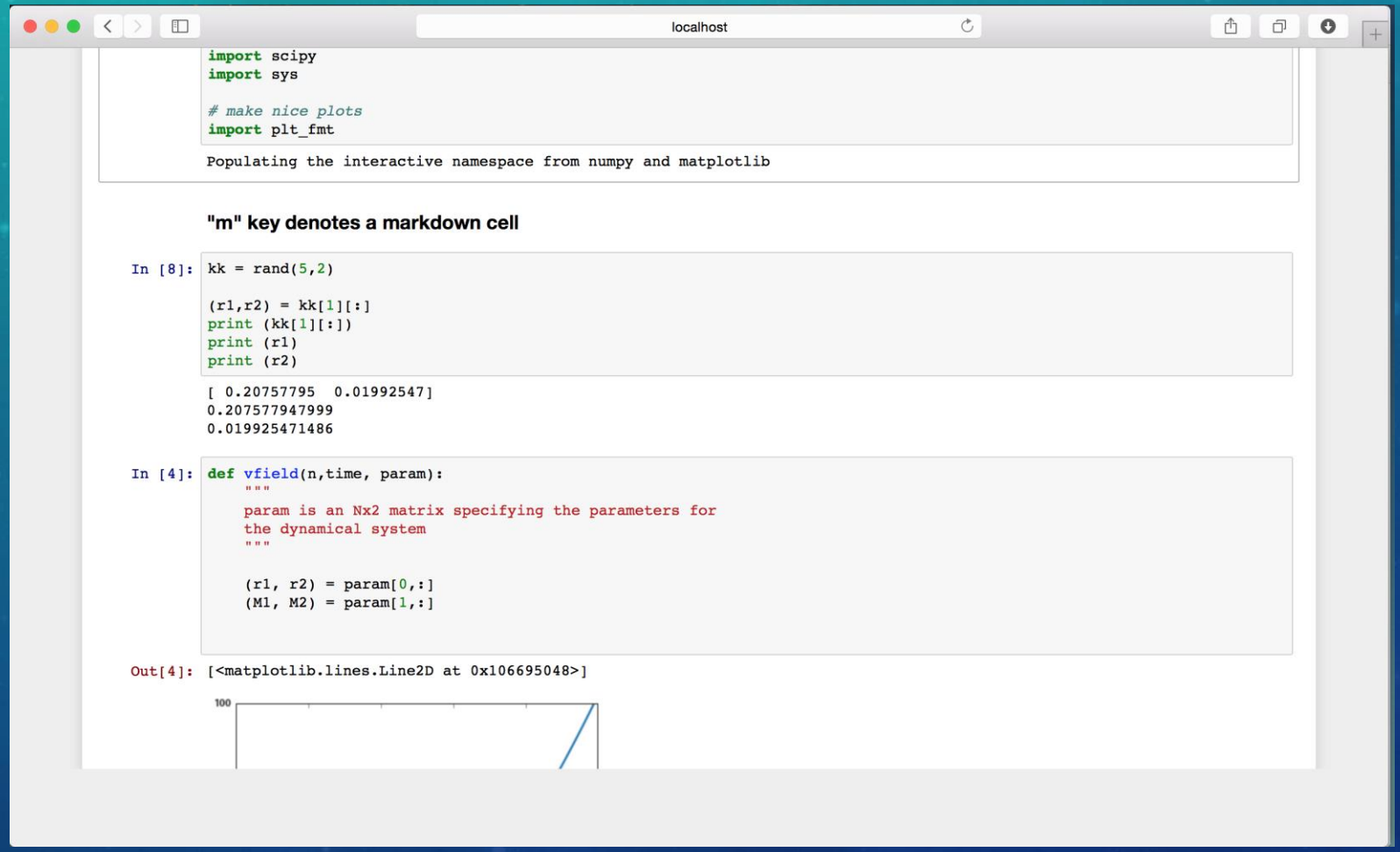



RMarkdown
LaTeX
knitr
R
PDF
HTML

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

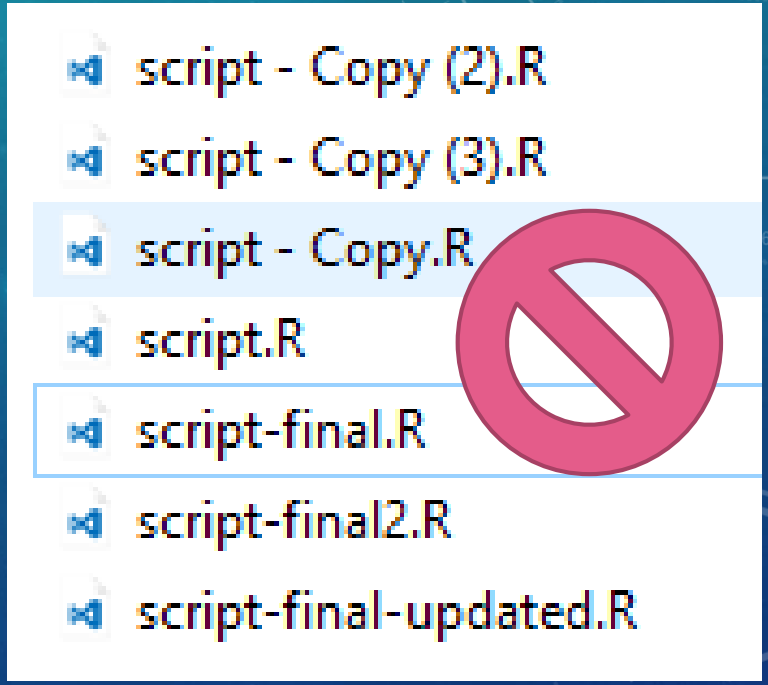


Jupyter Notebooks

- 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

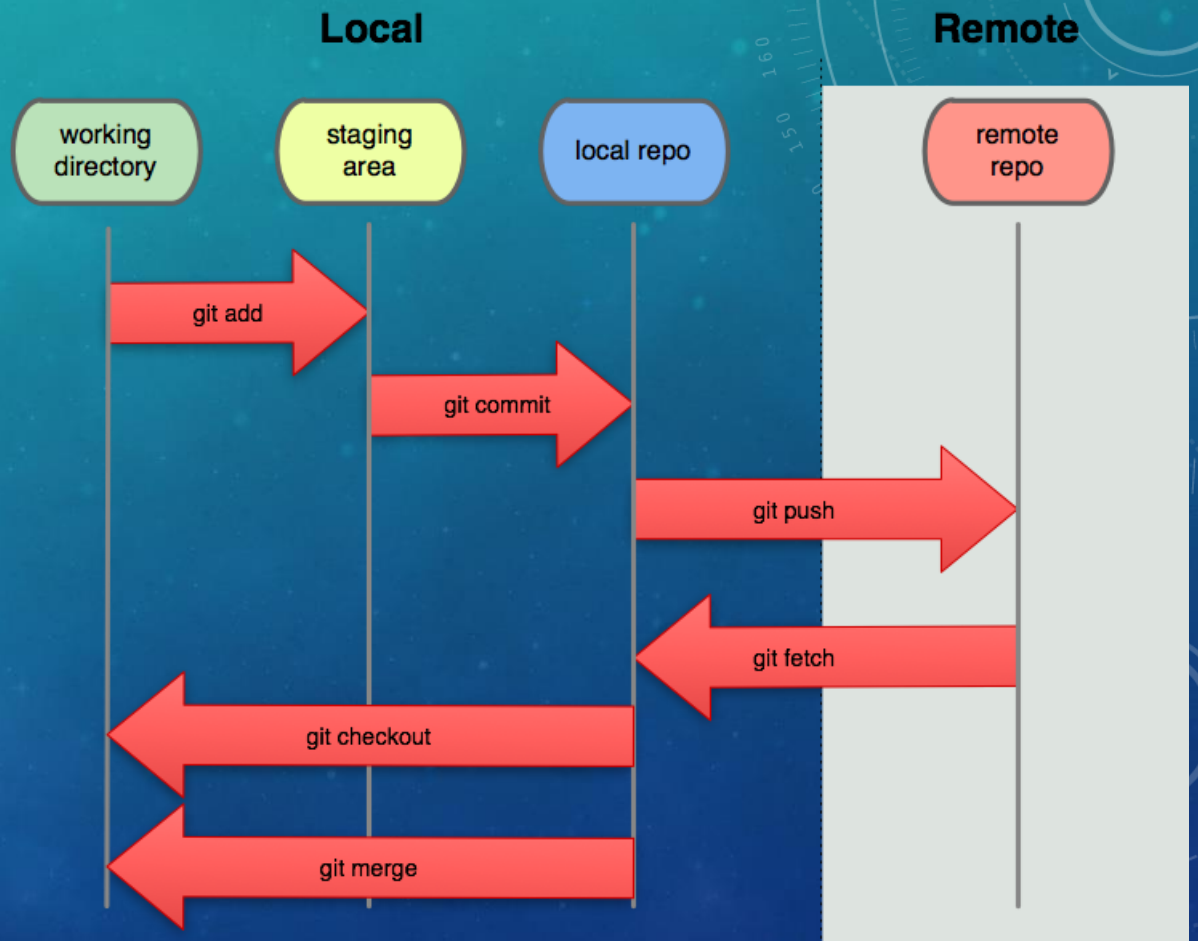
Version control

- 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 because 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

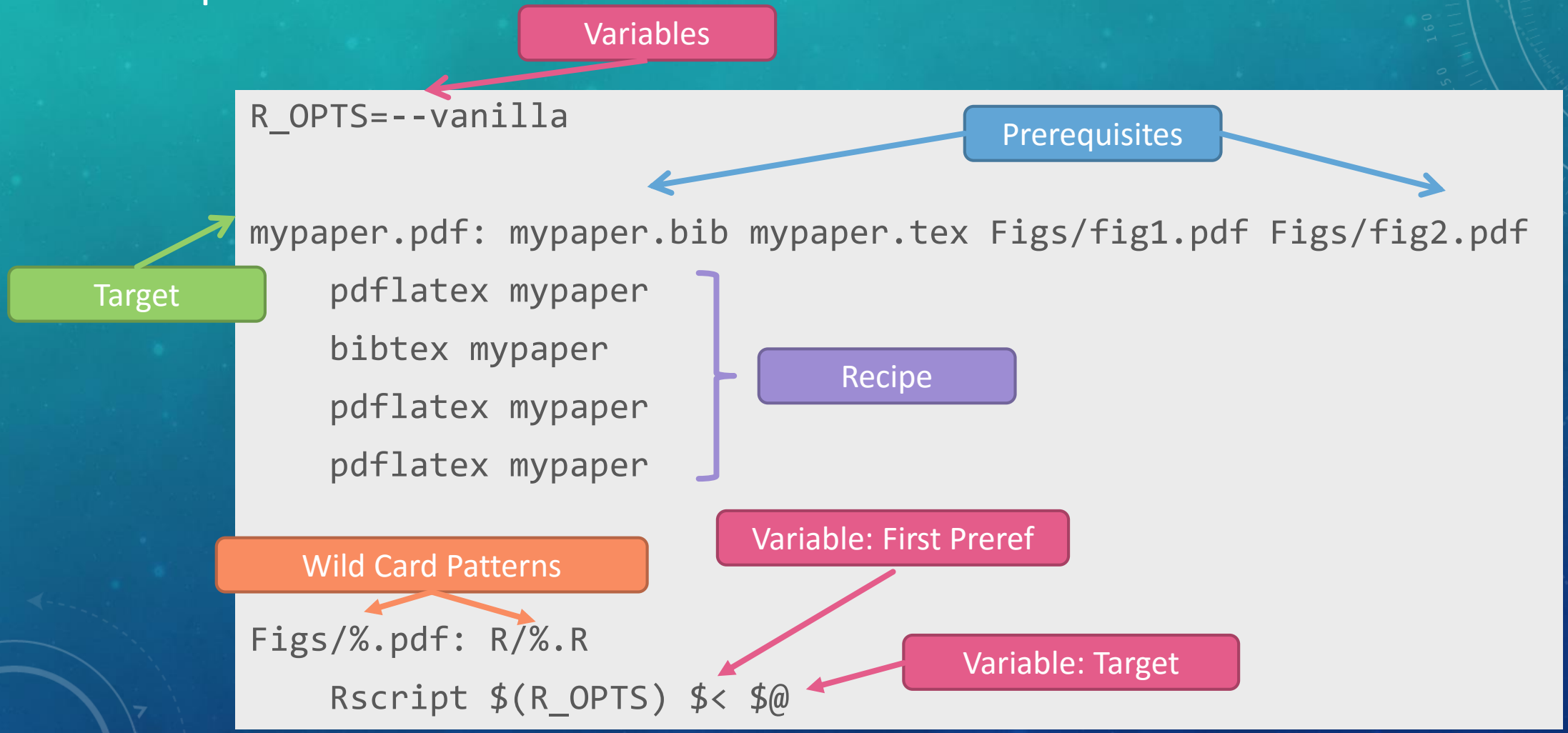
Execution automation

- 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 prerequsites

Sample make file



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

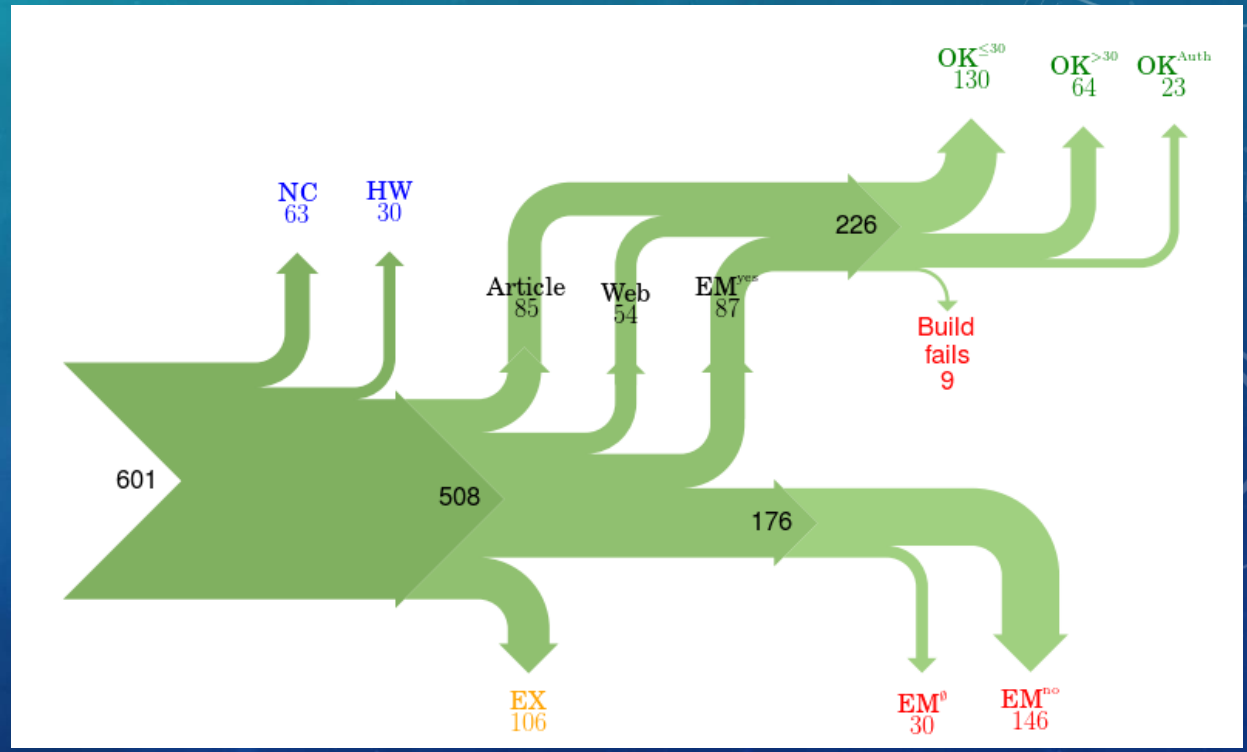
Reproducibility isn't easy

…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
- 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 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 piplines using make (https://bsmith89.github.io/make-bml/)

THANK YOU