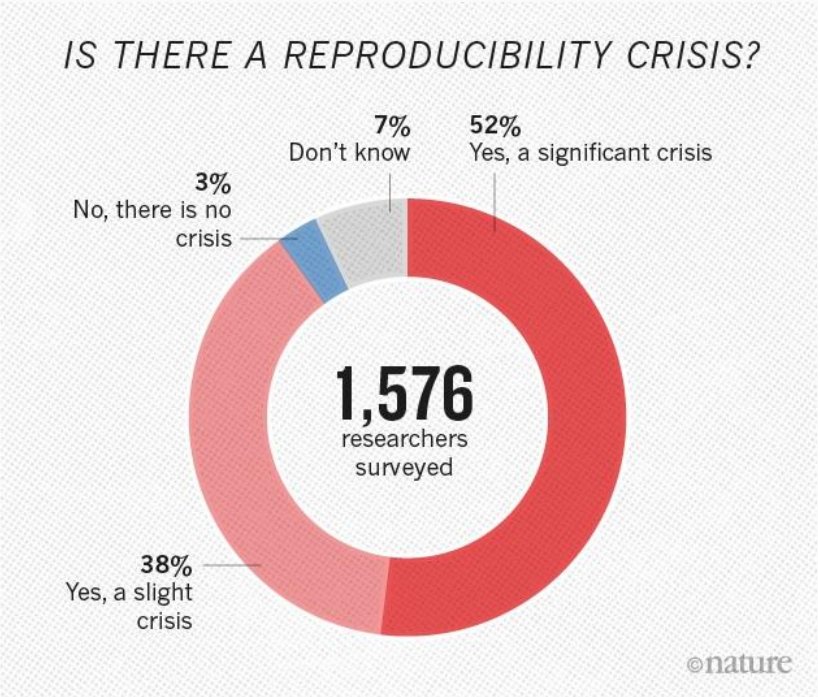
Reproducible Bioinformatics Research

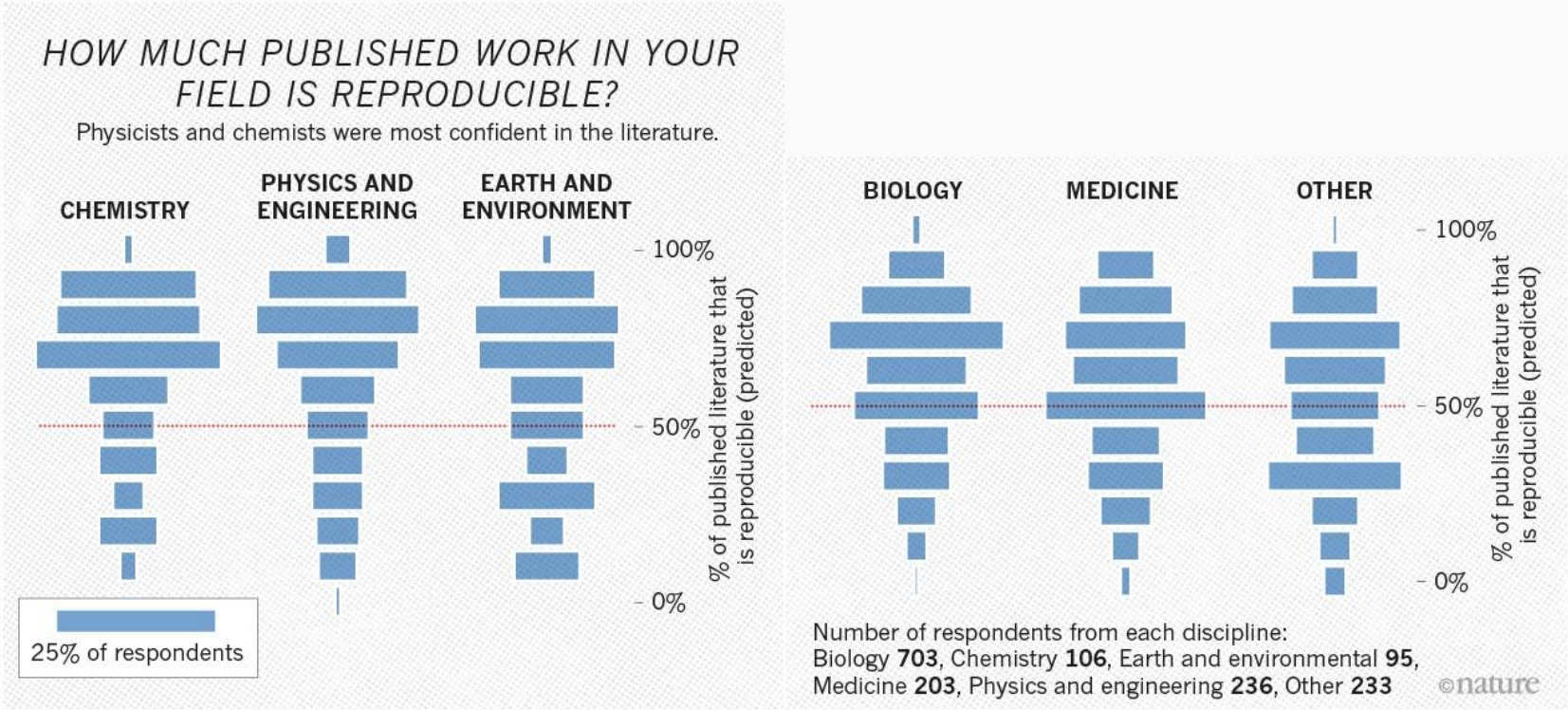
The Reproducibility “Crisis”

“More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments.”

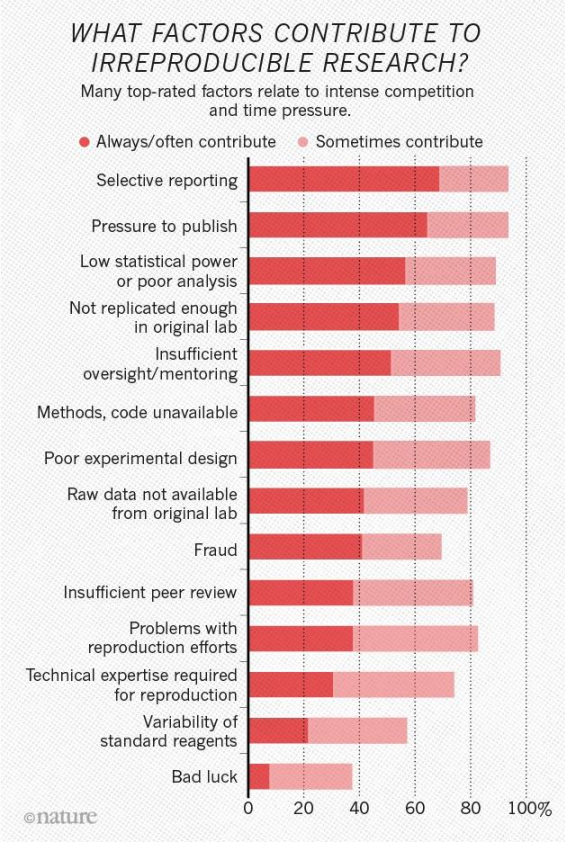


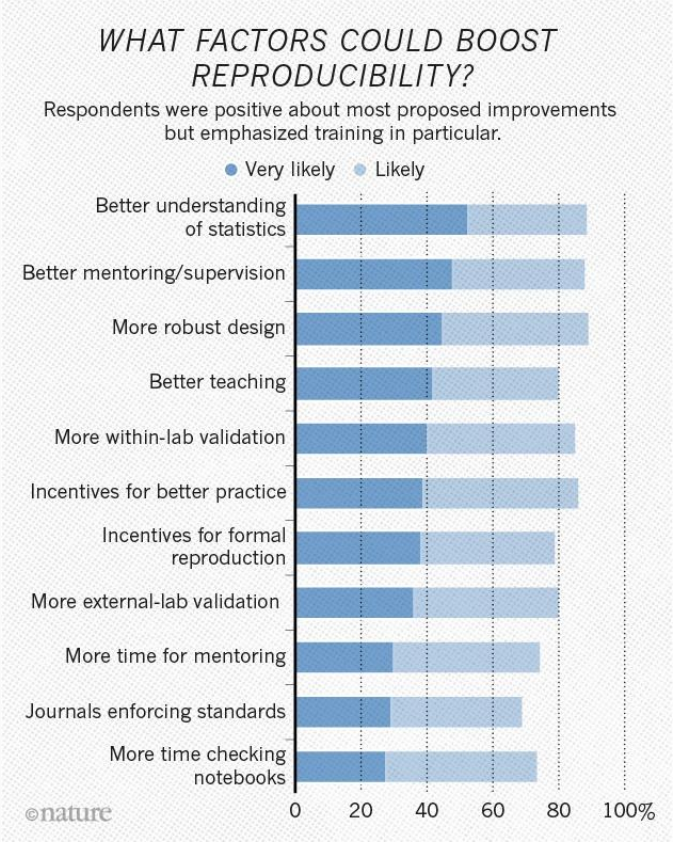
Baker, Monya. 2016. “1,500 Scientists Lift the Lid on Reproducibility.” Nature 533 (7604): 452–54.

The Reproducibility “Crisis”



The Reproducibility “Crisis”





Reproducibility vs Replicability

- **Reproduce:** create identical conditions to a previous result and come to the identical result
- **Replicate:** repeat a previously described experiment with new material or data

Depending on who you ask, these definitions are reversed!

http://languagelog.ldc.upenn.edu/nll/?p=21956

What is Reproducibility?

- **Reproduce:** create identical conditions to a previous result and come to the identical result
- For Bioinformatics, this means:
  - apply *the same methodology* (ideally same code) to
  - *the same data* and obtain
  - *the same result*
- If code is not available, must first **reimplement** the method

What is Replicability?

- **Replicate:** repeat a previously described experiment with new material or data
- For Bioinformatics, this means:
  - apply *the same methodology* to *new data* or
  - *a new methodology* to *the same data* and arrive at
  - *the same biological conclusion*
- If code is not available, must first **reimplement** the method

Definitions for this lecture

- **Reproduce** = show methodology was applied correctly
- **Replicate** = test whether the scientific result is “true”
- **Reimplement** = turn verbal description of method into a new reification of the method
- **Reapply** = run existing code on existing or new data

In summary:

Reproducibility and **replicability** are about *results*

Reimplementation and **reapplication** are about *methods*

Reproducibility in Bioinformatics

Four Main Components of Analysis

1. Computational Environment
2. Data
3. Code
4. Presentation

Kitchen Analogy

Code

\+

Environment

Data

Presentation

Computational Environment

Software Environment

- Software requires other software
- e.g. DESeq2 → bioconductor → R → OS libs
- Every piece of software has a *version*
- Not all OSes have all software versions

Must describe software *and* versions *and* their dependencies *and* their versions to fully recreate an environment!

Hardware Environment

- Analyses are run on specific hardware
- e.g. intel, AMD, GPU, SCC
- Some software is hardware-specific

Any known hardware specificities must be described to recreate an environment

Environment Management

- **Environment management** = organization and configuration of a set of software
- It is: a set of operations that make specific software executable
- It *may* be: a portable description of the software environment

GNU modules

- System for organizing software and its dependencies
- Typically used by system administrators
- A module manipulates your shell to include specific files and libraries
- *Not* a means to portable reproducibility!
- Strengths: software locally maintained, easy to use
- Disadvantages: manually maintained, requires sysadmin support and/or knowledge

Anaconda and miniconda

- anaconda: python distribution and environment management suite
- miniconda: just environment management suite
- create *environment* that installs *and describes* a set of packages with versions
- software organized into *channels*, e.g. bioconda
- **Strengths:** easy to use, good environment description
- **Disadvantages:** packages must already be in channels, when it fails it fails *hard*...

Containerization

docker

Containerization

- Complete environment description
- Essentially mini operatings systems
- Contain all specific software needed for an analysis
- Custom code can also be loaded into container
- Current technologies: docker and singularity
- Strengths: ideal strategy for encapsulating and sharing environments
- Disadvantages: requires sysadmin knowhow, docker has prohibitive technical requirements for SCC

Data

Three types of data

1. **Source data**
    a. short read datasets, microarrays, etc
2. **Support data**
    a. Reference genomes, gene annotations, etc
3. **Transformed data**
    a. Alignments, gene counts, etc.

Source Data

- Raw, unprocessed data is always preferred
  - E.g. untrimmed reads directly from sequencer
- Data should be deposited in a publicly available repository
  - E.g. GEO, dbGaP, figshare, zenodo
- Repository should have a plan for longevity
  - No personal servers!

Support Data

- Data used to condense, process, annotate, and interpret source data
- Most support data are maintained in persistent repositories with consistent formats
- If there is a persistent link, specify it!
- If not, download and store data yourself

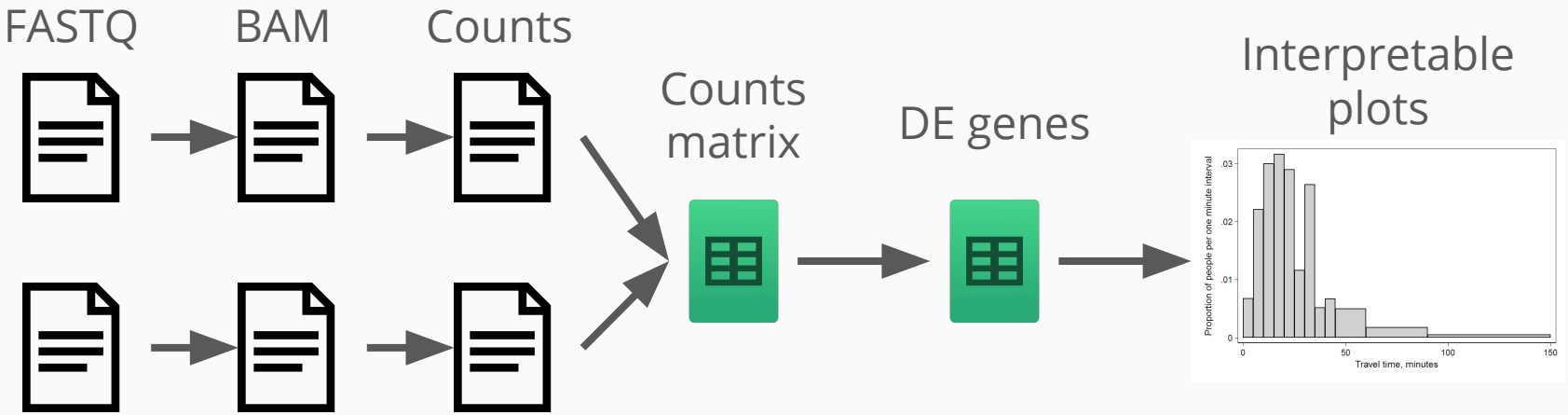
Transformed Data

- Derived from source(+support) data
- Usually what we use to interpret our results
- Code is a *recipe* for creating transformed data from source data
  - Should not be maintained as part of your workflow
- EXCEPT the final transformed form of the data used to make interpretations

Code

Code is a recipe

- Describes how data is transformed from one form to another
- Combination of *analysis* code and *glue* code



Good Coding Practices

- Code changes drastically during development
- It is tempting to make copies of scripts to preserve old versions
  - v1, v2, v2.1, v2.1_good, v2.1_final, v3...
- Version Control Systems, e.g. git, are far superior

Web VCS platforms

- Web platforms (github.com, bitbucket.org) enable:
  - Seamless backup of code
  - Simplified sharing and collaboration
  - Public dissemination of analysis code upon publication
- Employers look for your public software repos!

Code Documentation

- Code should be well documented in comments and README files, also:
- Code should be well documented in comments and README files, then:
- Code should be well documented in comments and README files, also then:
- Future you will thank current you for it

Presentation

Tables and Figures

- Main vehicles of scientific communication
- Guide readers through manuscripts
- Visualizing data is very powerful, important,
- ...and very challenging
- *ALL* the data underlying a figure must be available in textual form as well

Tables and Tabular Data

- *Avoid copy and paste whenever possible!*
- Tabular data should (almost) always be included as supplementary materials
- Standardize formats (CSV, not excel!)
- Human- and machine-readable:
  - consistent , controlled textual values
  - column headers, no comment rows
  - no irregular formatting, etc

Figures

- Create figures programmatically from transformed data whenever possible
- Invest in learning plotting libraries (matplotlib, seaborn, ggplot, etc)
- Output to Scalable Vector Format (SVG) rather than bitmap formats