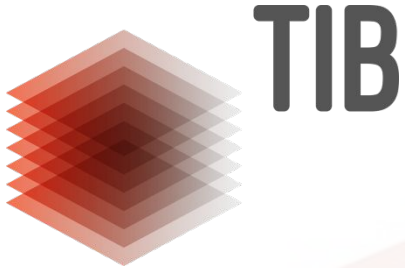
LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY



TIB

Interoperable

Konrad Förstner, Luke Johnston, Mateusz Kuzak, Katrin Leinweber

TIB, 11. July 2018 Recording: doi.org/10.5446/37826
FAIR Data & Software (Carpentries-based workshop) **#TIBFDS**

Leibniz Association

Interoperability Agenda

1. **Definitions, roles & TIB's vision of knowledge-based information flows**

2. Very basic practices for data & software management projects

3. for software: "play well with others"
   - Using Python modules & creating one
   - (Using R packages & building one)

to be Interoperable

I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation

I2. (meta)data use vocabularies that follow FAIR principles

I3. (meta)data include qualified references to other (meta)data

Your institution’s / repository’s role

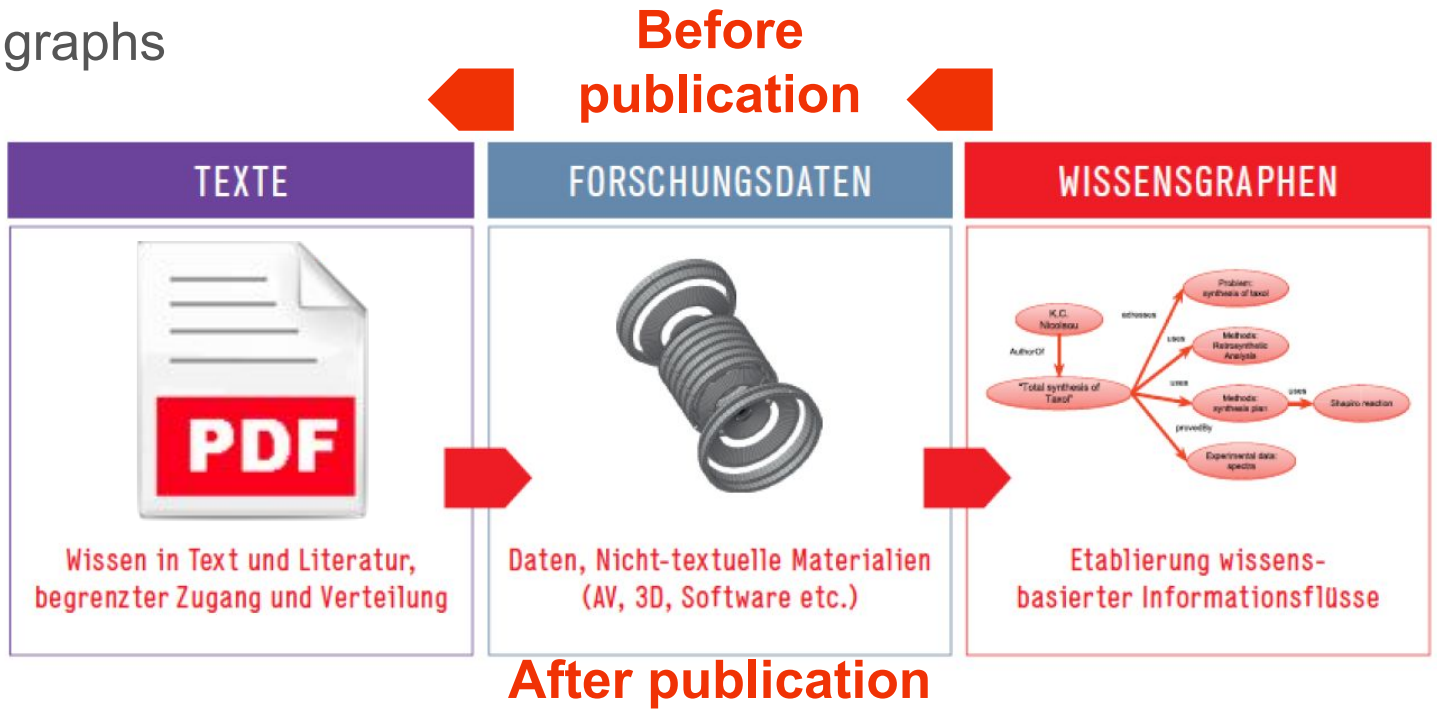
- provide machine-readable (meta)data with a well-established formalism
- structured, using discipline-established vocabularies / ontologies / thesauri (RDF extensible knowledge representation model, OWL, JSON LD, schema.org)
- support referencing metadata fields between datasets via a schema (relatedIdentifer, relationType)
- offer (meta)data ingest from relevant sources (Document Information Dictionary or Extensible Metadata Platform from PDF)

Your role as a scientist

- provide as precise & complete metadata as possible
- look for metrics to evaluate the FAIRness of a controlled vocabulary / ontology / thesaurus
  - often do not (yet) exist
  - assist in their development
- clearly identify relationships between datasets in the metadata (e.g. “is new version of”, “is supplement to”, “relates to”, etc.)
  - request support regarding these tasks from the repositories in your field of study
- for software: follow established code style guides (thanks to @npch!)

Vision: Research will move from document- to knowledge-based information flows

- semantic descriptions of research data & their structures
- aggregation, development & teaching of subject-specific vocabularies, ontologies & knowledge graphs



Vision: Research will move from document- to knowledge-based information flows

several TIB groups are working towards this

- Data Science and Digital Libraries **=>** (research) knowledge graph(s)
- Scientific Data Management
- Visual Analytics to expose information within videos as keywords => av.tib.eu
- Scientific Knowledge Engineering **=>** ontologies

PIDs provide interoperable Metadata

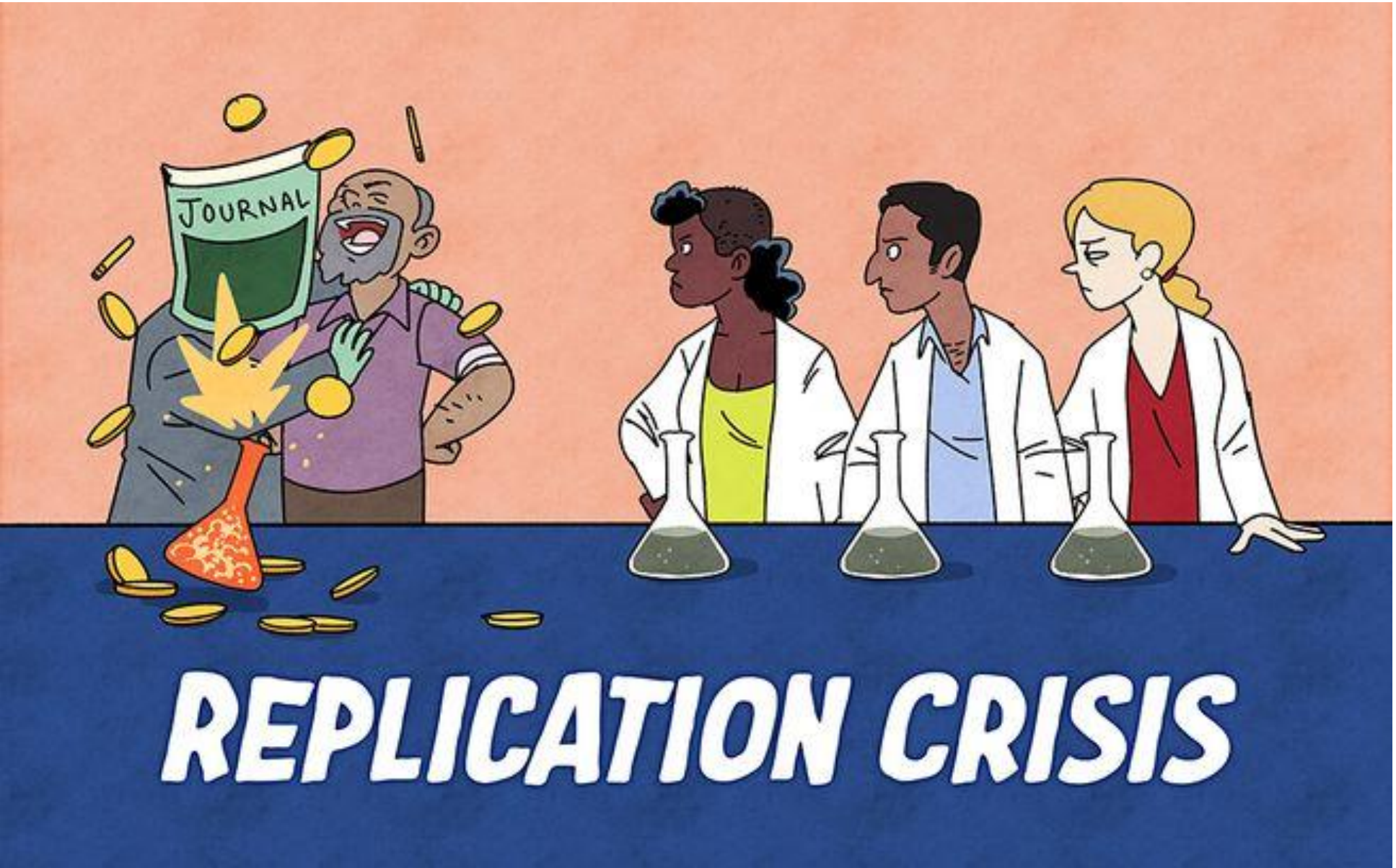
- Example:

→ Automatic ORCID profile update when DOI is minted

DataCite – CrossRef – ORCID collaboration

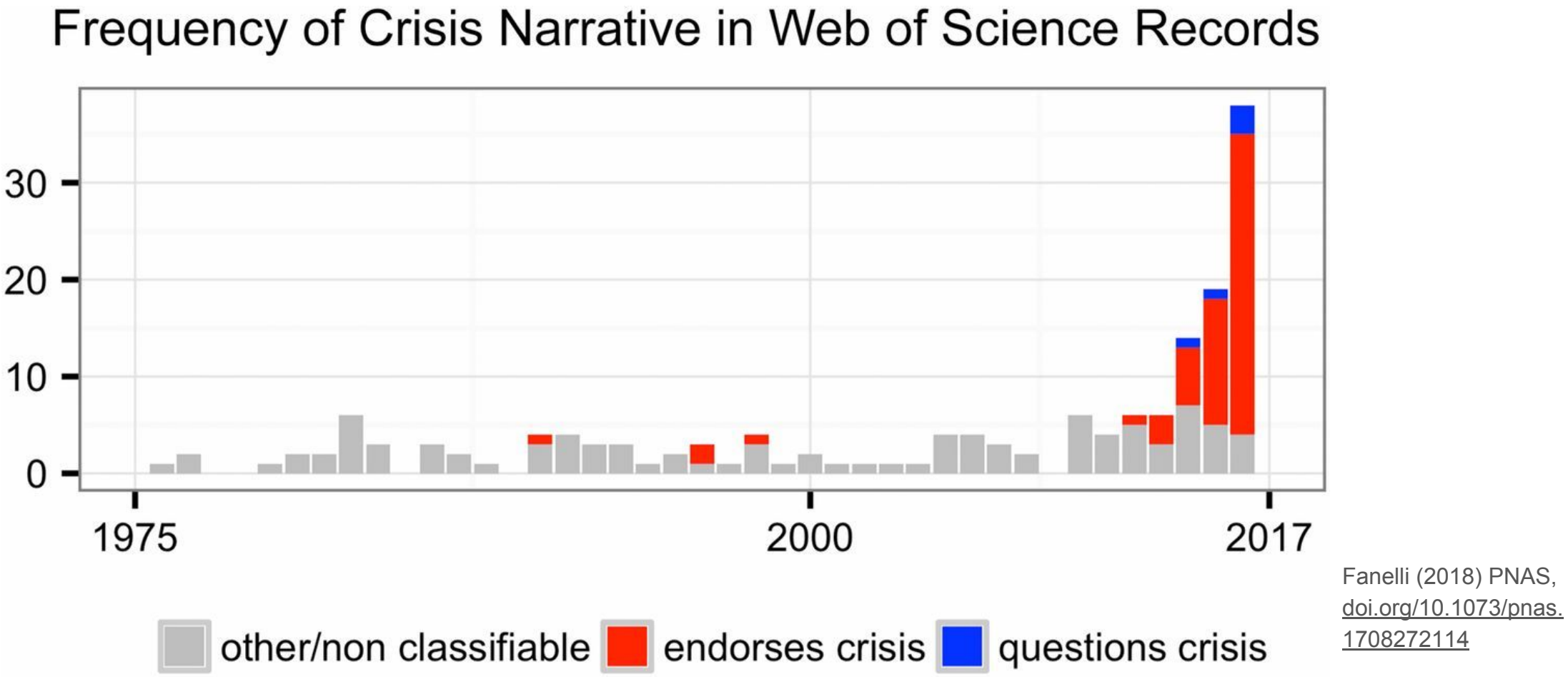
→ PID of choice for RDM:
Here: The Digital Object Identifier (DOI)

Detour: Replication / Reproducibility Crisis



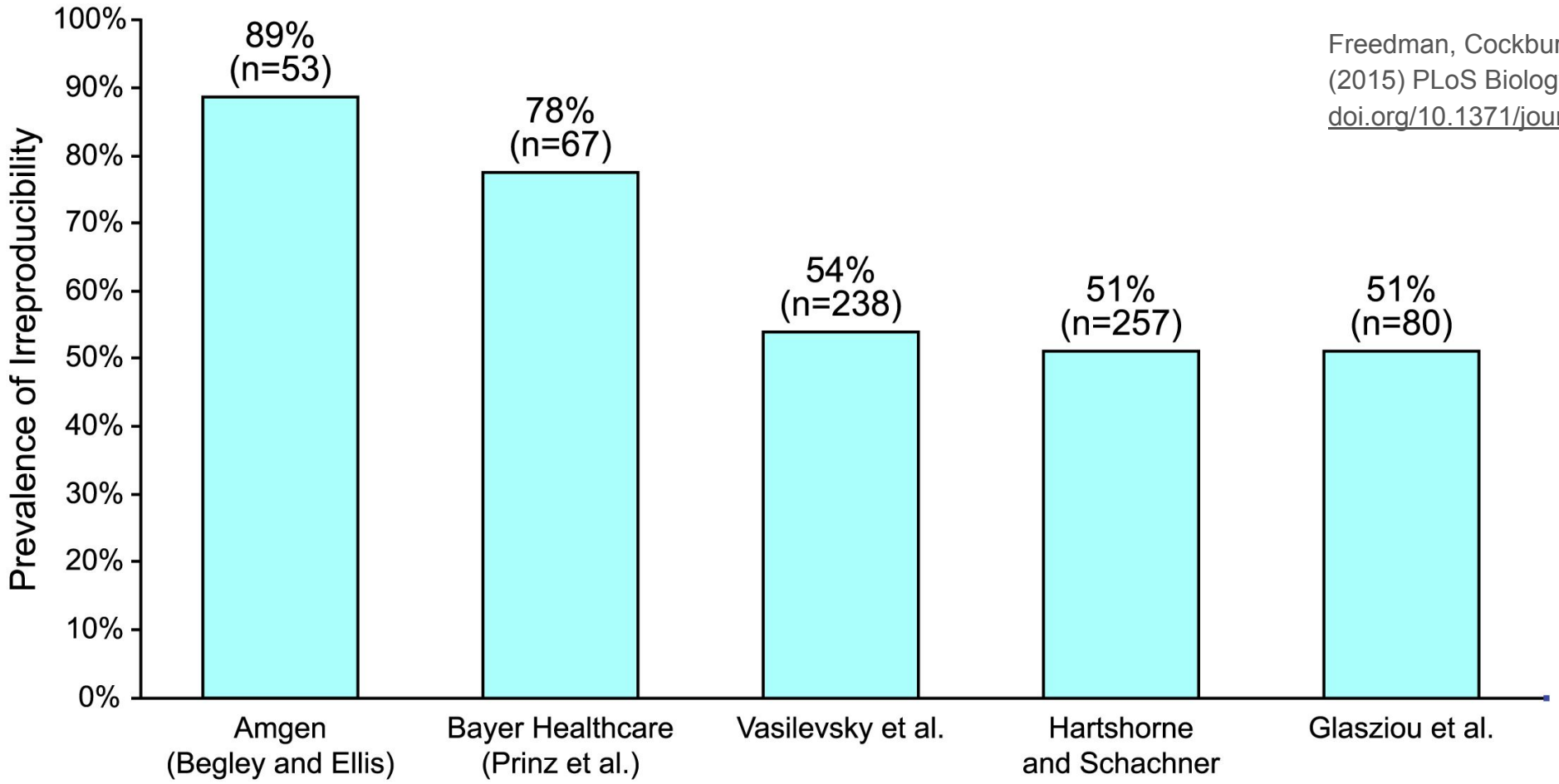
© Maki Naro
makinaro.com

Detour: Replication / Reproducibility Crisis



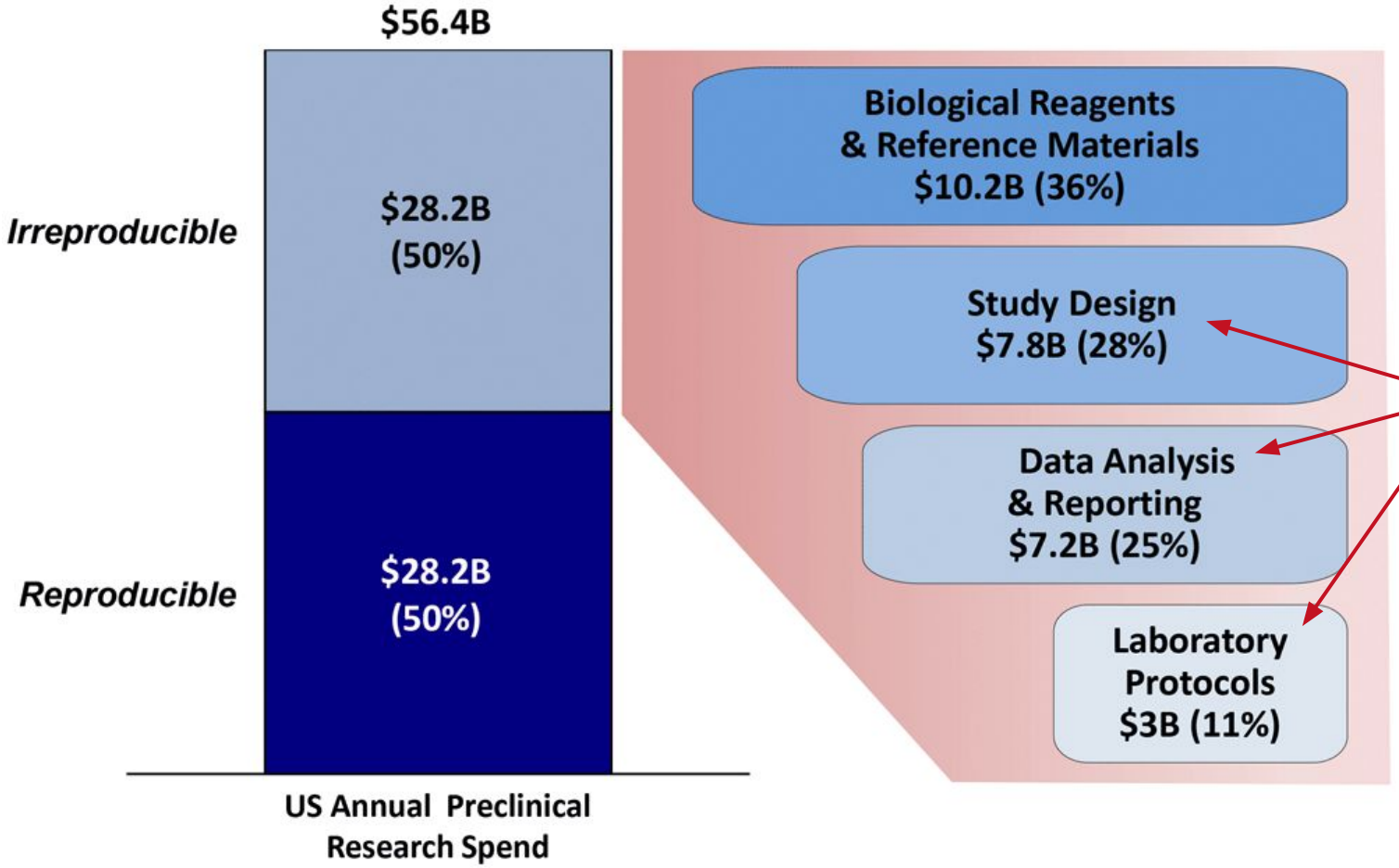
Fanelli (2018) PNAS, doi.org/10.1073/pnas.1708272114

Detour: Replication / Reproducibility Crisis leaves bad impression



Freedman, Cockburn & Simcoe (2015) PLoS Biology, doi.org/10.1371/journal.pbio.1002165

Detour: Replication / Reproducibility Crisis



Freedman, Venugopalan & Wisman (2017) F1000Research, doi.org/10.12688/f1000research.11334.1

may contain traces of software

Examples of science failing due to software errors/bugs: figshare.com/authors/Neil_Chue_Hong/96503

Interoperability Agenda

1. Definitions, roles & TIB's vision of knowledge-based information flows

2. **Very basic practices for data & software management projects**

3. for software: "play well with others"
   - Using Python modules & creating one
   - (Using R packages & building one)

Organising files & folders in a project ("cookiecutter")

A) Organized by File type

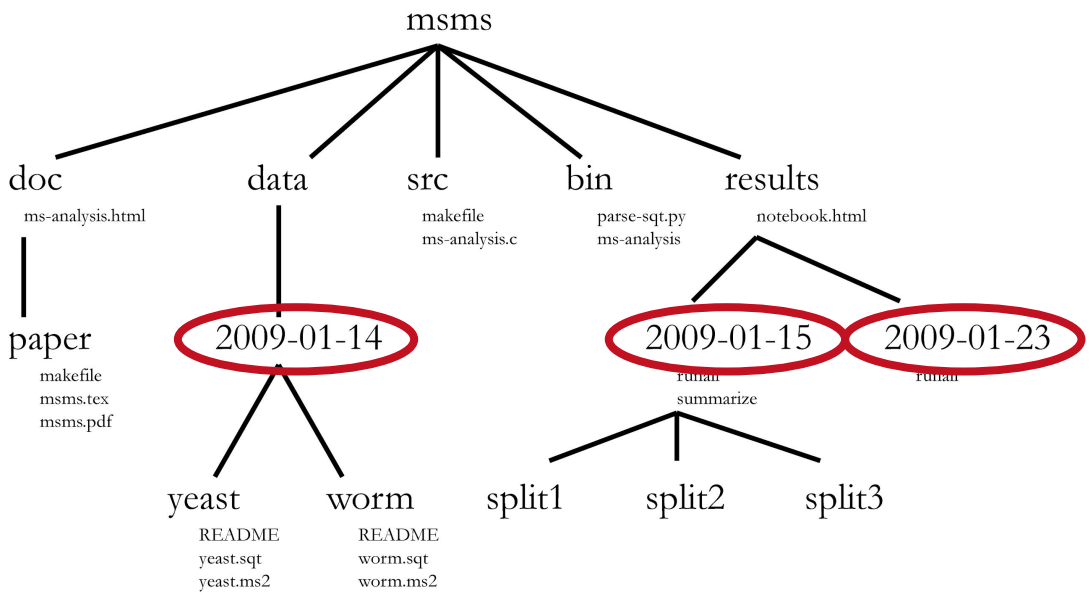
Example.A
- Code
  - Step.1
  - Step.2
- Data
  - Processed
  - Raw
- Results
  - Figure.1
  - Figure.2
  - Models
- readme.txt

B) Organized by Analysis

Example.B
- Figure.1
  - Code
  - Data
  - Results
- Figure.2
  - Code
  - Data
  - Results
- readme.txt

dataDryad.org/pages/reusabilityBestPractices (CC-BY 3.0)

- repository might suggest a scheme
- if not, look in your community of practice
- **stick to an established convention**
- **version control enables risk-free changes**
- **Date format? ISO 8601: YYYY-MM-DD!**



Noble WS (2009) A Quick Guide to Organizing Computational Biology Projects. PLoS Comput Biol 5(7): e1000424. doi:10doi.org/10.1371/journal.pcbi.1000424

- larger project rather use object(-based) storage
  - docs.google.com/presentation/d/1Ma-hctXcE6AkqdfoHSFrEH9W98V9pnjrav3Zz3DFSrc has unique ID, to which metadata & content are attached regardless of logical or physical location (drive, directory, etc.)

Interoperability Agenda

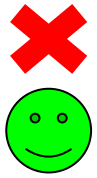
1. Definitions, roles & TIB's vision of knowledge-based information flows

2. Very basic practices for data & software management projects

3. for software: "play well with others"
   - Using Python modules & creating one
   - (Using R packages & building one)

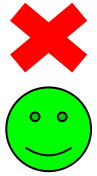
Basic rules for interoperable scripts

- load modules / packages / etc. explicitly atop the file: `import … as …` & `library('…')`
- hard-coding absolute folder paths results in errors for anyone else
  - instead: relative paths within the organised project folder (see above)



`numpy.loadtxt(fname='/Users/YOU/project-X/data/inflammation-01.csv')`

`numpy.loadtxt(fname='../data/inflammation-01.csv')` **or** `__file__`



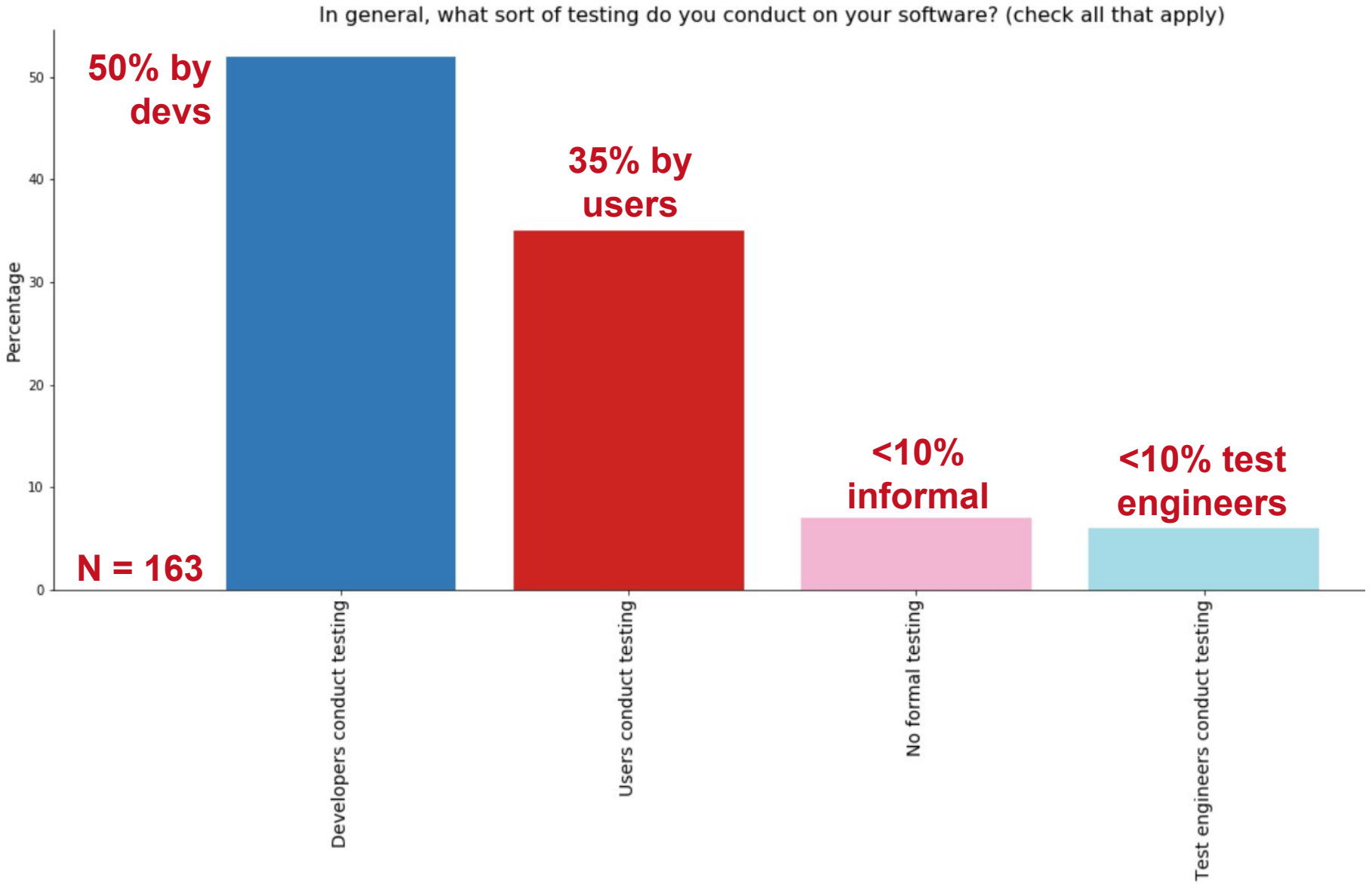
`setwd("C:\Users\YOU\path\that\nobody\else\has")`

`.rproj` files

from: Jenny Bryan (2018) Project-oriented workflow & more tips on Twitter.com/HadleyWickham/status/940021008764846080

Better yet: build a module / package!

Testing software, preferably automatically



"[...] around 70% of research relies on software [...] if almost a half of that software is untested, this is a huge risk to the reliability of research results."

Results from a US survey about Research Software Engineers URSSI.us/blog/2018/06/21/results-from-a-us-survey-about-research-software-engineers (Daniel S. Katz, Sandra Gesing, Olivier Philippe, and Simon Hettrick)

Olivier Philippe, Martin Hammitzsch, Stephan Janosch, Anelda van der Walt, Ben van Werkhoven, Simon Hettrick, Daniel S. Katz, Katrin Leinweber, Sandra Gesing, Stephan Druskat. 2018. doi.org/10.5281/zenodo.1194669

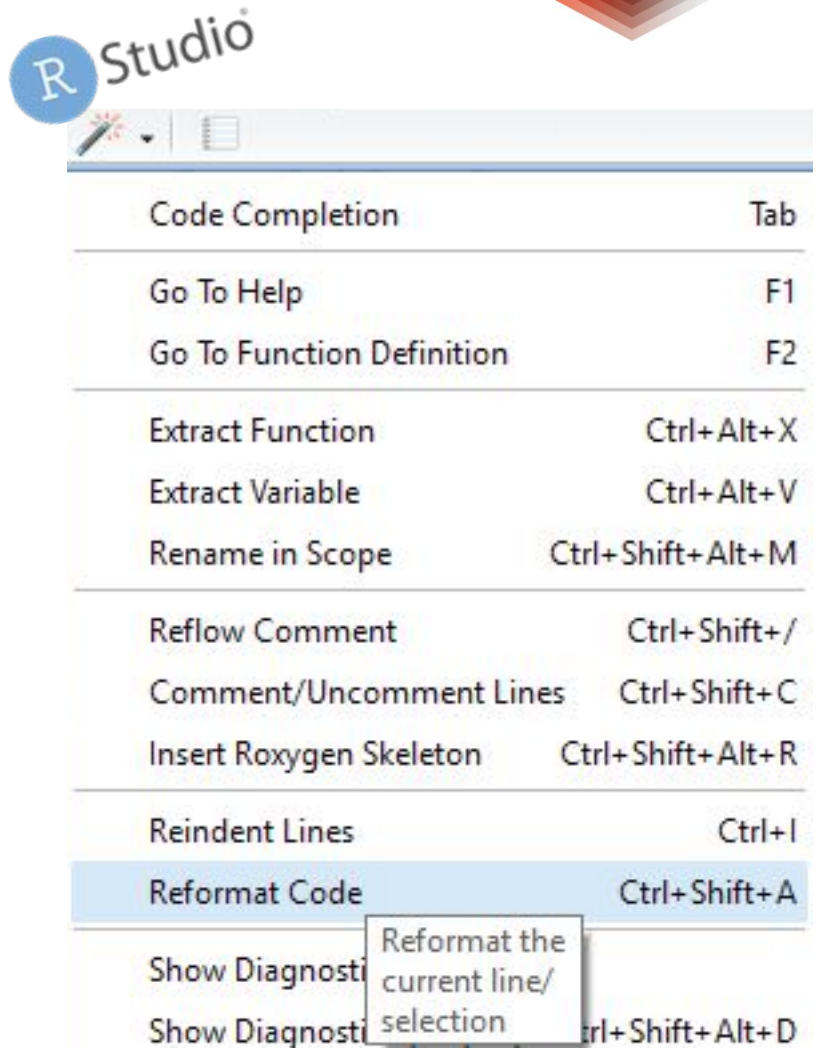
Code style guides & formatters (thanks to Neil Chu Hong)

- faster than manual/menial formatting
- code looks the same, regardless of author
  - can be automated enforced to keep diffs focussed

- PyPI.org/project/pycodestyle, /black, etc.

- ROpenSci packaging guide
- style.tidyverse.org

- Google.GitHub.io/styleguide



What's in a version number? Semantics!

- If others **can** use your code, convey the meaning of updates with SemVer.org (CC BY 3.0)
  - "version number[changes] convey meaning about the underlying code" (Tom Preston-Werner)

v 0 . 4 . 2

major (0. => 1.)
removed or renamed functions

"I need to update my own code!"

minor
(.4. => .5.)
backwards-compatible additions or changes

"If I don't need new stuff, I can ignore this."

patch (.2 => .3)
backwards-compatible bug-fixes

"I can install without updating my code."

- Value outcome for user(s) more than output by developer(s)!
- live coding now: Python & TIBHannover.GitHub.io/FAIR-R

LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY

TIB

Thanks to Konrad Förstner :-)

Which questions do you have for us?

Contact information:
Katrin.Leinweber@TIB.eu & Angelina.Kraft@TIB.eu
T +49 511 762-14693 & -14238

Leibniz Association