LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY

TIB

Summary of Key Points

Hammitzsch, Johnston, Kuzak, Kraft, Leinweber

TIB, 13. July 2018 Recording: doi.org/10.5446/37828
FAIR Data & Software (Carpentries-based workshop) **#TIBFDS**

Leibniz
Association

Summaries

1. **FAIR principles**
2. General remarks
3. Additional references, recommended resources, etc.
4. Survey

On the FAIR principles

- **Open consultation on FAIR data action plan** of the European Open Science Cloud expert group (deadline 5 Aug 2018):
  GitHub.com/FAIR-Data-EG/Action-Plan - feel free to voice your opinion!

- FAIR does not mean open (data)

- There are different degrees of FAIRness, as research disciplines, resource types (e.g. data and software) and their requirements are strongly varied - but the shared goal is quality & good scientific practice

- FAIR (in its origins) focuses first and foremost on machine to machine interactions, only secondary on human to machine (or human to human) interactions

- DMPs/SMPs and a licence help to keep data/software FAIR

- Depending on the discipline, the researchers background and legal aspects: "Partly FAIR may be FAIR enough"

- REMEMBER: 'As open as possible, as closed as necessary'

THE CARPENTRIES

merged in 2017

software carpentry

DATA CARPENTRY

up & coming

library carpentry

∀uthorCarpentry

High-Performance Computing

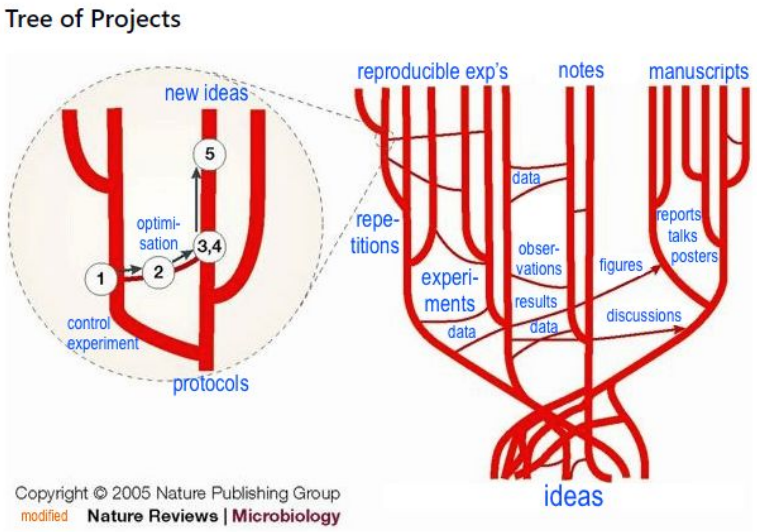
- teach computing & data skills to scientists
  - things that “work too well to be worth teaching” doi.org/10.12688/f1000research.3-62
  - youtu.be/1e26rp6qPbA?t=35s
- collaborative, community-developed lessons: bio- & genomics, geospatial, social, ...
- evidence-based pedagogical methods (sources)
  - instructor training & mentoring
  - learning-by-doing & live coding

Interested in setting up a local group?

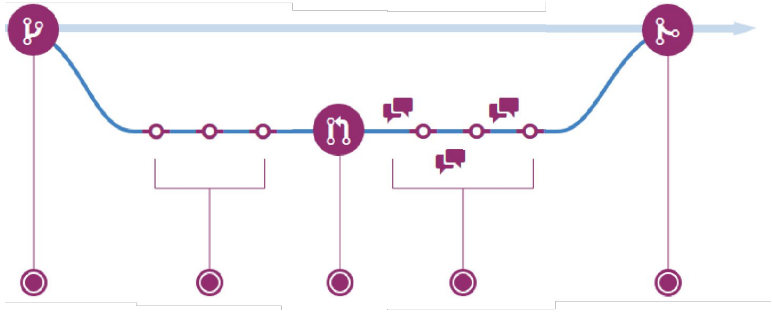
Please contact Carpentries@TIB.eu!

Accessible projects through public Git repos

1. **VC** because **self-documenting/-explaining** a project's evolution
2. easier **publication** & opportunity for **collaborations**
   a. Pull/Merge Requests = workflow for 21st century => fearless change, smooth review workflow, automation potential
   b. **issue tracker, website hosting, project management, etc.**
3. GitHub + Zenodo.org = DOI

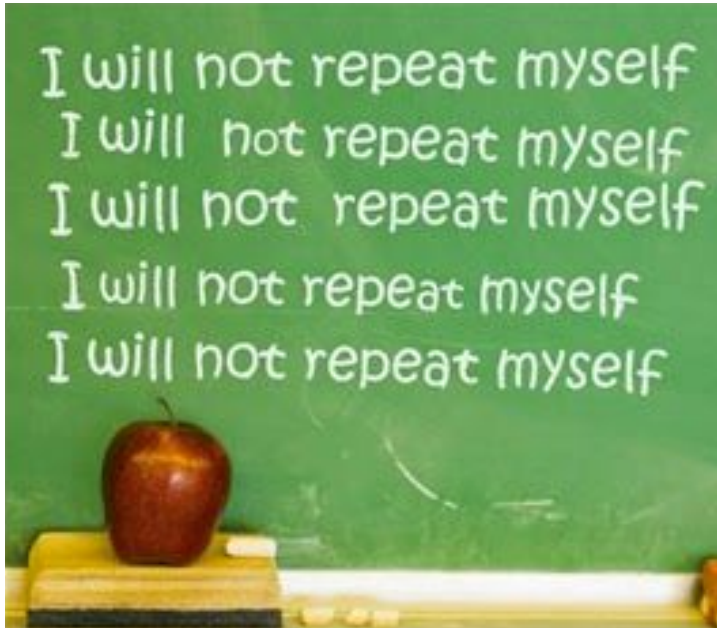


Git repo can be single source of truth & communication channel



guides.GitHub.com/introduction/flow

Accessible, interoperable & reusable code through functions



- scripts without code duplication may not need functions
- but instead of copy-pasting: stay DRY, not WET
  en.Wikipedia.org/?curid=3291957
- code is a liability / technical debt

"Programs are meant to be read by humans and only incidentally for computers to execute."

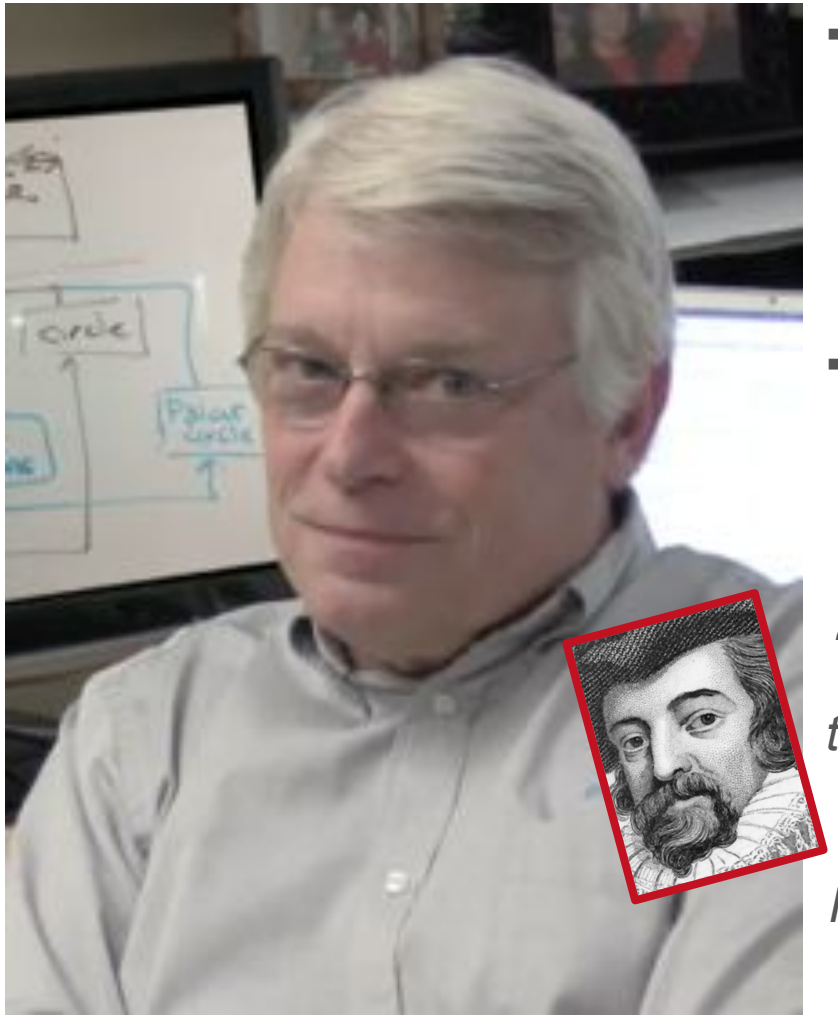
Donald Knuth (inventor of TeX)
flickr.com/vonguard (cropped; CC-BY-SA-2.0)

- concise function names are the beginning of good documentation
- individual & automatic "unit testing"
- more comprehensive approach: packaging!

Summaries

1. FAIR principles
2. General remarks
3. Additional references, recommended resources, etc.
4. Survey

Refactoring (software hygiene)



by Tim-bezhashvyly (CC BY-SA 4.0)

- changing code internally, without changing its behaviour
  - renaming functions, variables, etc.
  - extracting duplicate code into new functions
  - re-organising code, files, etc.

- tests remove the fear from this process
  - automation & quick feedback make this fun
  - exercism.io/languages/r & /python

"You don't go fast by rushing [but by] taking your time, studying the problem, thinking carefully, cleaning up." -- Robert C. Martin

*First make it work, **then make it right**, and, finally, make it fast.*

-- Stephen C. Johnson and Brian W. Kernighan

“... finally, make it fast”. If it’s *really* necessary:

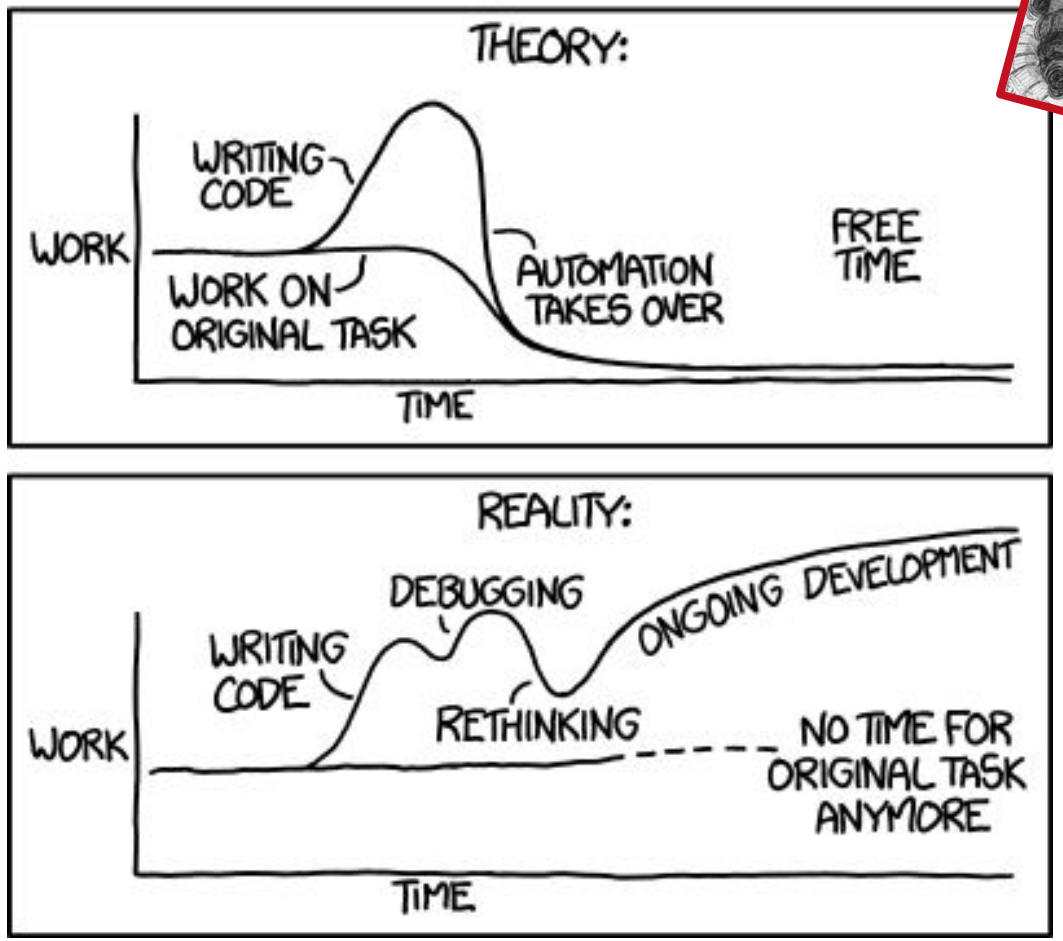
1. **Profile / benchmark**
2. **Identify bottleneck (under your control?)**
3. **(If yes) Remove**
4. **Profile / benchmark again**
5. **Identify next bottleneck**
6. **...**

wiki.python.org/moin/PythonSpeed

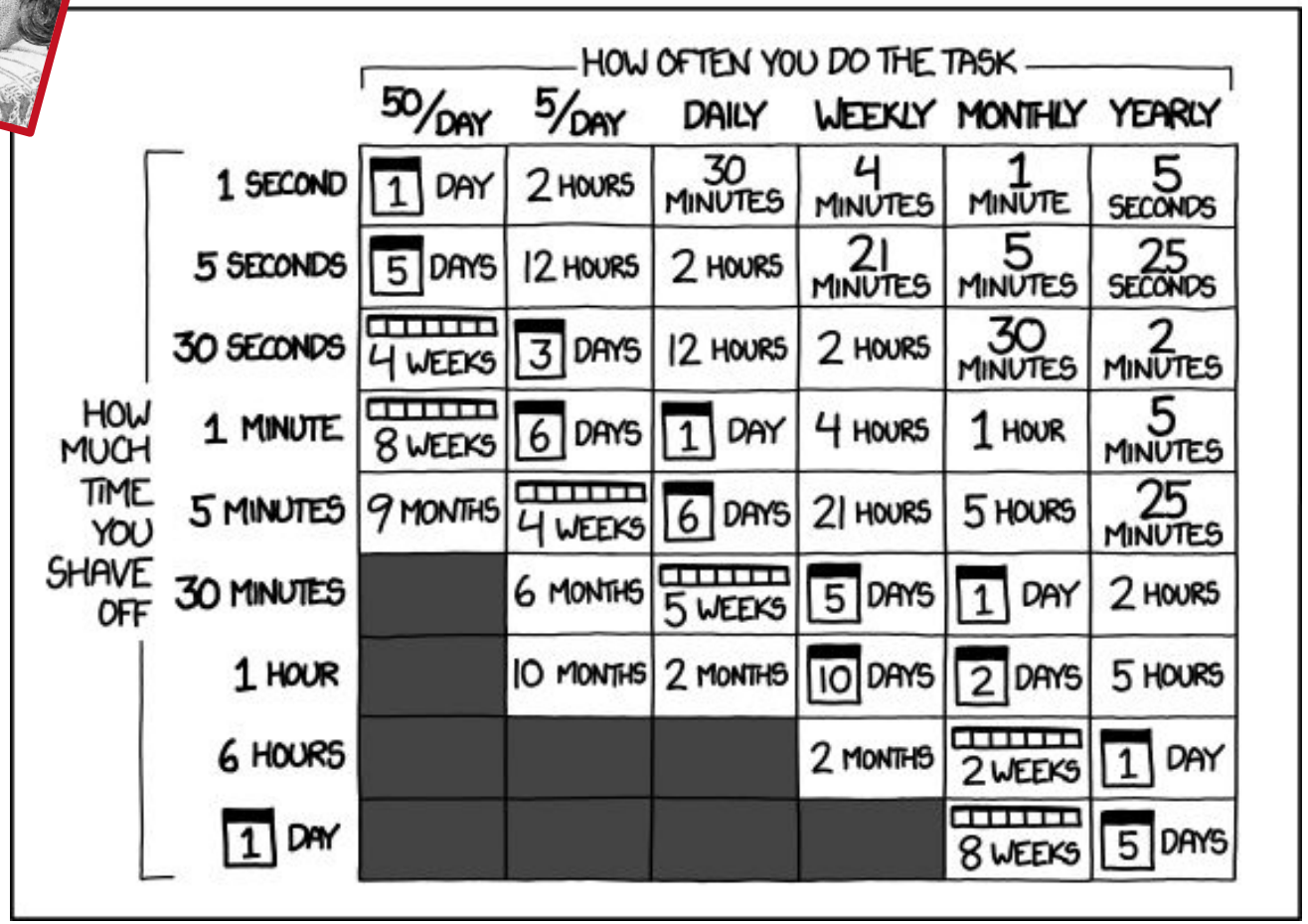
TGMStat.WordPress.com/2013/09, RStudio.GitHub.io/profvis & Adv-R.had.co.nz/Profiling.html

- for measuring individual functions: CRAN.R-project.org/package=microbenchmark

“... finally, make it fast”. A word of warning...



xkcd.com/1319

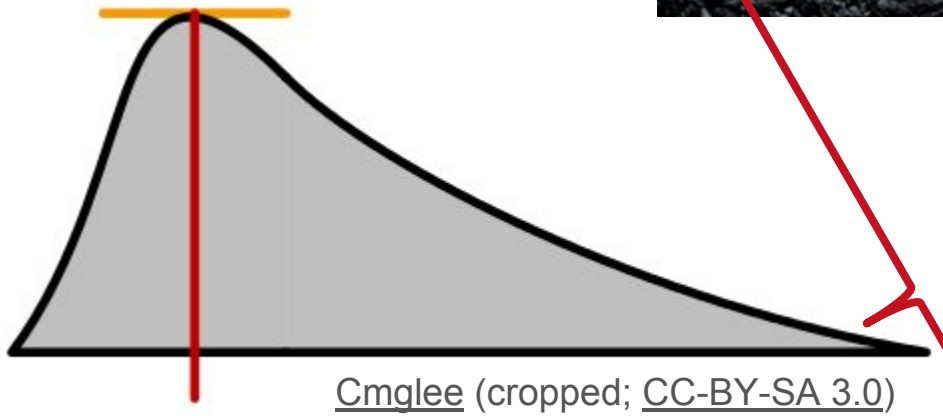


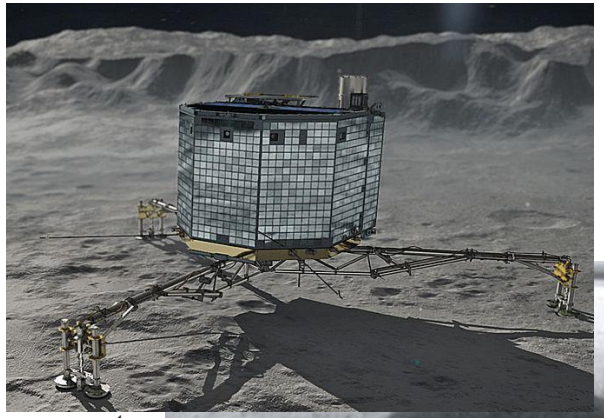
xkcd.com/1205

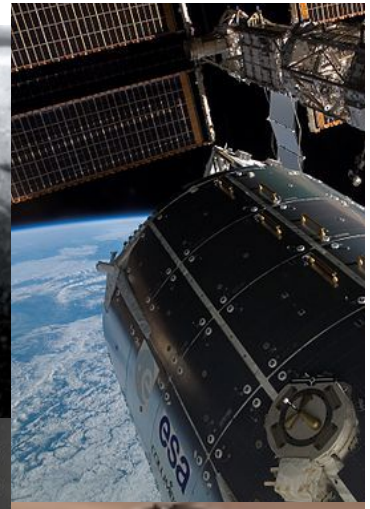
One more thing about “best practices”...

- By the time any “practice” becomes widely recognised as “best”, it’s average
- more = harder to choose
  - “best practices in scientific computing” doi:10.1371/journal.pbio.1001745
  - “good enough practices” doi:10.1371/journal.pcbi.1005510
  - “4 simple recommendations” doi:10.12688/f1000research.11407.1
  - “10 recommendations” doi:10.1186/2047-217X-3-31
  - etc....

Pick one, follow it and evaluate next level at next project start.







Summaries

1. FAIR principles
2. General remarks
3. Additional references, recommended resources, etc.
4. Survey

Resources that help you find licences & info about them

a certain *type* of licence

ifrOSS .org (German)

news about court cases

Just tell me one...

ChooseAlicence.com (GitHub)

UFAL.GitHub.io/public-licence-selector/

`summary(licences)`

tldrLegal.com

I want to read more...

opensource.guide/legal

scholar.Google.com/scholar?q=Stodden

Morin et al. (2012) ... licencing for the Scientist-Programmer. doi.org/0.1371/journal.pcbi.1002598

doi.org/10.1371/journal.pcbi.1005510

Most important: pick one! For software, "MIT, BSD, or Apache [are] good enough"

Additional references, recommended resources, etc.

- General: guide.eScienceCenter.nl & OpenSource.guide
- Conference videos! But beware of “hype-driven development”...
  - R-project.org/conferences.html, RStudio.com/conference & RStudio.com/resources/webinars
    - GitHub.com/RStudio/webinars
  - conference.SciPy.org & YouTube.com/user/EnthoughtMedia/playlists
  - YouTube.com/user/GotoConferences/playlists (IT development & management)
- Blogs
  - blog.GitHub.com/category/education
  - about.GitLab.com/blog/categories/open-source & /releases
  - suistainable software.ac.uk/blog
- GitHub.com/awesomedata/awesome-public-datasets
- on technical debt & software sustainability
  - ivory.idyll.org/blog/2017-pof-software-archivability.html & ivory.idyll.org/blog/2018-oss-framework-cpr.html
  - Gael-Varoquaux.info/programming/software-for-reproducible-science-lets-not-have-a-misunderstanding.html
  - nadiaeghbal.com/tragedy-of-the-commons
- UNIX philosophy of software development: catb.org/~esr/writings/taoup/html/ch01s06.html
- building larger software systems: laputan.org
- practice more: programminghistorian.org, Software & Data Carpentry lessons
  - bio-informatics problems: rosalind.info

Additional references, recommended resources, etc.

- textbooks
  - Effective Computation in Physics (physics.codes)
  - Advanced R, R for Data Science, R packages & more: hadley.nz
  - teaching: Carpentries.GitHub.io/instructor-training/reference.html#books
- courses: OpenScienceMOOC.GitHub.io
- about automatic software tests and TDD: WatirMelon.blog/testing-pyramids, blog.cleancoder.com,
  - TDD in many languages: exercism.io

- try.GitHub.io,
- reading: Think-like-a-git.net,

- our R lesson: TIBHannover.GitHub.io/FAIR-R
- package source & guides: ROpenSci.org & CRAN.R-project.org
- naming R packages: njtierney.com/post/2018/06/20/naming-things/ & GitHub.com/ROpenSciLabs/available
- packaging & reproducible reports: GitHub.com/BenMarwick/rrtools

LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY

TIB

Thanks to Konrad Förstner, Luke Johnson, Mateusz Kuzak and Martin Hammitzsch!

Leibniz
Association

LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY

TIB

Farewell!

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

Leibniz
Association