LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY

TIB

Accessible

Angelina Kraft, Konrad Förstner, Katrin Leinweber, Luke Johnston

TIB, 10. July 2018 Recording: doi.org/10.5446/37825
FAIR Data & Software (Carpentries-based workshop) #TIBFDS

Leibniz
Association

Accessibility Agenda

1. **Definitions & roles**

2. Version control & project management with Git(Hub)

3. Lunch

4. “Accessible” software ≈ “comprehensible” code
   - Functions in Python & R

For data to be Accessible

A1. (meta)data are retrievable by their identifier using a standardized communications protocol

- A1.1 the protocol is open, free, and universally implementable
- A1.2 the protocol allows for an authentication and authorization procedure, where necessary

A2. metadata remain accessible, even when the data are no longer available

Your institution's / repository's role

Accessible

- as little access barriers as necessary (user accounts for sensitive data)
- metadata contains contact and responsibility information
- open communication protocols: HTTPS, SFTP, e-mail, SSH, git
  - warning sign: special up-/download tool (justified for large volumes of data)
- offer "tombstone" pages for datasets that had to be deleted
- have exit strategy for (meta)data preservation

Your role as a scientist

Accessible

- access data programmatically whenever possible (web services + R packages & Python modules)
- e-mail requests sometimes necessary & also FAIR (sensitive data)
  - if granted: secure access possible? password manager => unique passwords!
- metadata: can help to plan research (esp. replication)
- request these features from the repositories in your field of study

Accessibility Agenda

1. Definitions & roles

2. Version control & project management with Git(Hub)

3. Lunch

4. “Accessible” software ≈ “comprehensible” code
    - Functions in Python & R

for Software to be Accessible

A1. (meta)data are retrievable by their identifier using a standardized communications protocol

A1.1 the protocol is open, free, and universally implementable

A1.2 the protocol allows for an authentication and authorization procedure, where necessary

A2. metadata remain accessible, even when the data are no longer available

git Mercurial Subversion

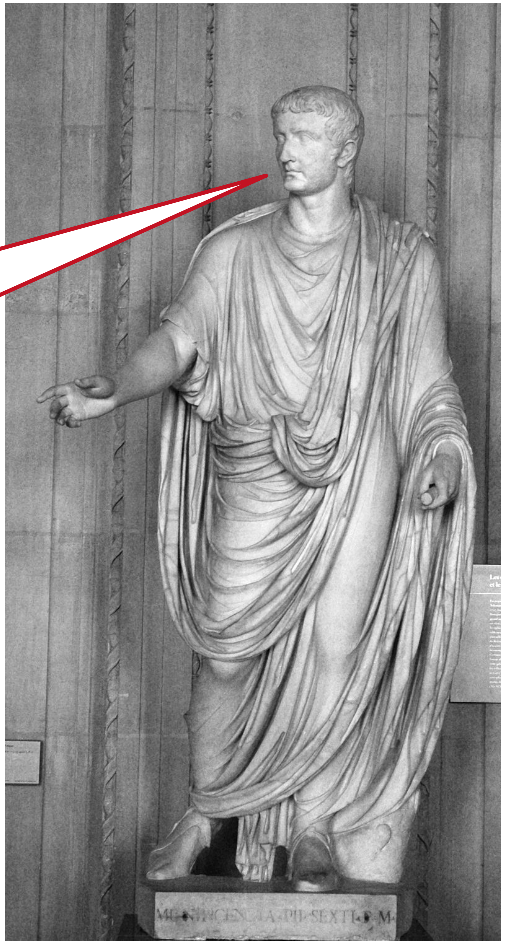
- IDs: commit hashes, repo/project URLs
- GitHub: topics & description

- Microsoft's CodePlex.com & code.Google.com went read-only; GitHub will be bought by MS
- SoftwareHeritage.org as secondary source

Recap from yesterday: GitHub + Zenodo = DOI

- guides.GitHub.com/activities/citable-code
- official integration thanks to science.Mozilla.org/projects/codemeta
- intrinsic IDs (Git’s SHA1 hashes) vs. “minted” PIDs
  - technical vs. procedural persistence
- Zenodo: file backup & persistent landing page for each release version
  - demo: GitHub.com/TIBHannover/BacDiveR/issues/14
- look for “Software Deposit and Preservation Workshop” on

  Software.ac.uk/tags/software-preservation



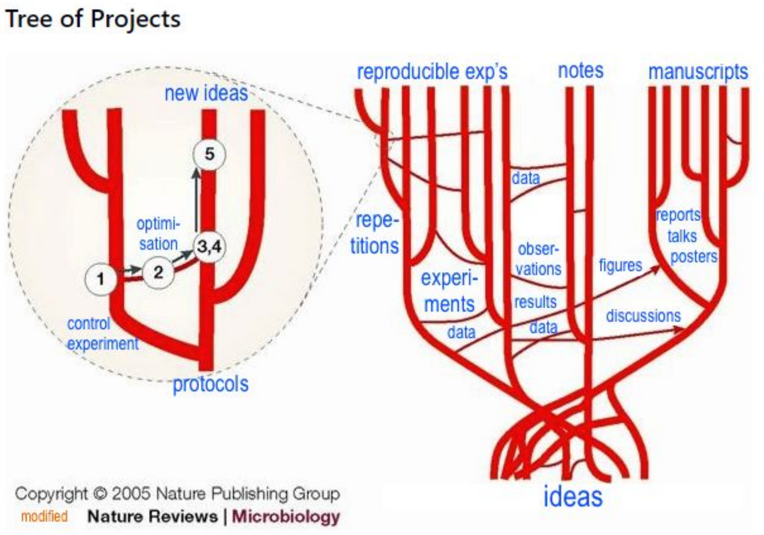
Public Domain by Marie-Lan Nguyen, via commons.Wikimedia.org/w/index.php?curid=549920

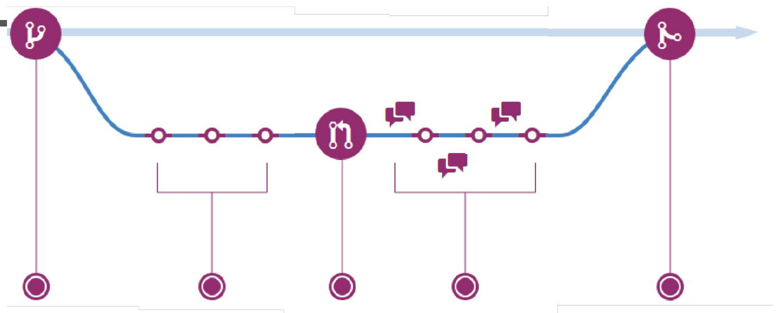
Why version control? Why Git(Hub/Lab)?

1. **VC** because **self-documenting/-explaining** a project’s evolution
2. **Git** because of strongest **network effects**
3. easier **publication** & opportunity for **collaborations**
   a. *“[...] addition of Windows support ... requested for many years [but] ignored ... because none of [us had] experience with Windows. John Cary ... implemented the needed changes ... in just one weekend!” -- Sherry Li (bssw.io)*
   b. BacDiveR *mea culpa*: started privately => “scooped by” BacDive
   c. Pull/Merge Requests = workflow for 21st century => fearless change, smooth review workflow, automation potential
   d. **issue tracker, website hosting, project management, etc.**

   => Git repo as a single source of truth

Leinweber (2018) Einführung in Git(Hub/Lab) av.tib.eu/media/32224

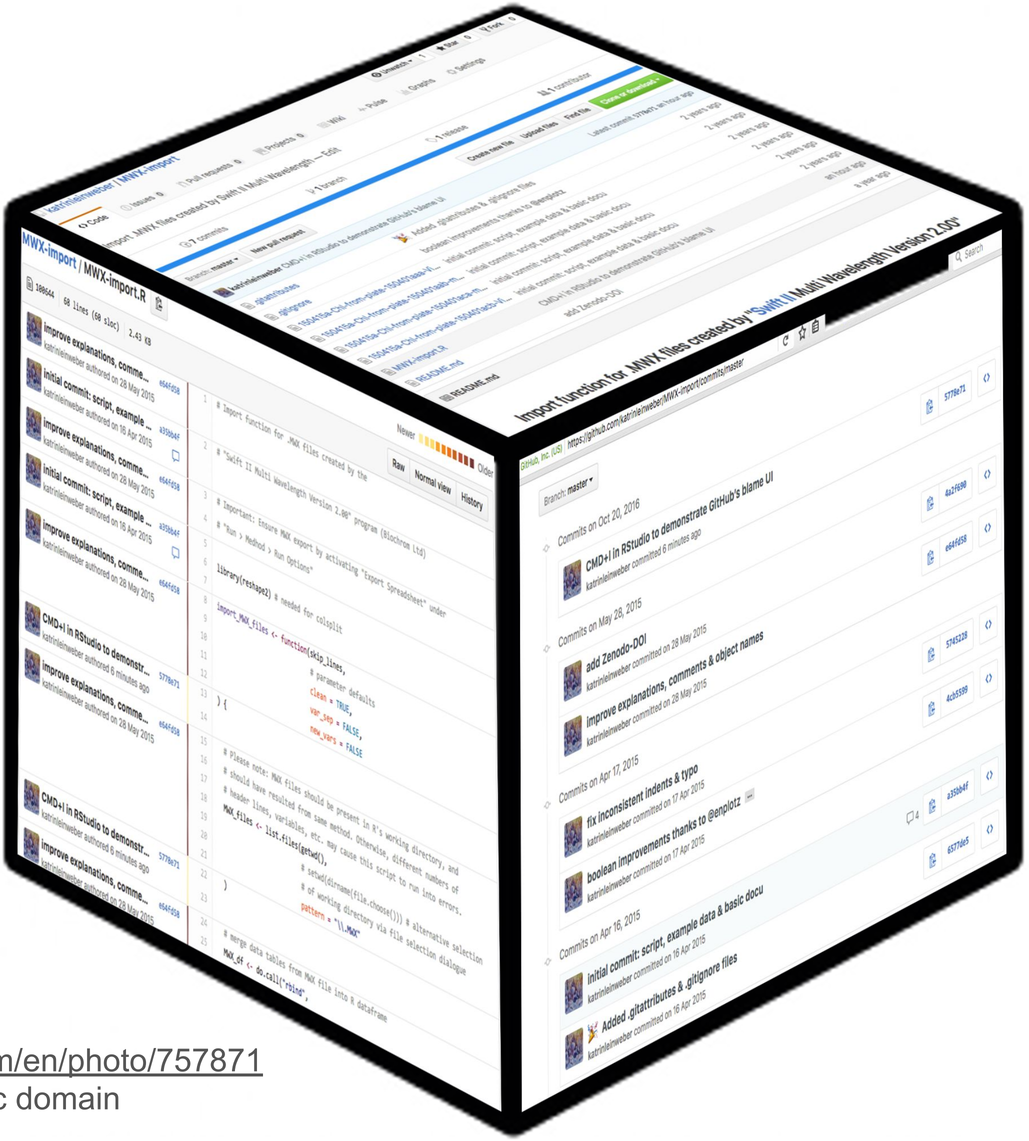




guides.GitHub.com/introduction/flow

Accessing the project’s evolution

- in 3D: normal file/folders, history / `log` & `blame` views
- microscope: `git show ...`
- time-machine: `git checkout ...`



katrinleinweber / MWX-import

Import .MWX files created by Swift II Multi Wavelength — Edit

katrinleinweber CMD+I in RStudio to demonstrate GitHub's blame UI ... Latest commit 5778e71 an hour ago

.gitattributes	Added .gitattributes & .gitignore files	2 years ago
.gitignore	boolean improvements thanks to @enplotz	2 years ago
150415a-Chl-from-plate-150401aaa-Vi...	initial commit: script, example data & basic docu	2 years ago
150415a-Chl-from-plate-150401aab-m...	initial commit: script, example data & basic docu	2 years ago
150415a-Chl-from-plate-150401aca-m...	initial commit: script, example data & basic docu	2 years ago
150415a-Chl-from-plate-150401acb-Vi...	initial commit: script, example data & basic docu	2 years ago
MWX-import.R	CMD+I in RStudio to demonstrate GitHub's blame UI	an hour ago
README.md	add Zenodo-DOI	a year ago

README.md

MWX-import / MWX-import.R

100644 | 68 lines (60 sloc) | 2.43 KB

improve explanations, comme... katrinleinweber authored on 28 May 2015	e64fd58
initial commit: script, example ... katrinleinweber authored on 16 Apr 2015	a35bb4f
improve explanations, comme... katrinleinweber authored on 28 May 2015	e64fd58
initial commit: script, example ... katrinleinweber authored on 16 Apr 2015	a35bb4f
improve explanations, comme... katrinleinweber authored on 28 May 2015	e64fd58
CMD+I in RStudio to demonstr... katrinleinweber authored 6 minutes ago	5778e71
improve explanations, comme... katrinleinweber authored on 28 May 2015	e64fd58
CMD+I in RStudio to demonstr... katrinleinweber authored 6 minutes ago	5778e71
improve explanations, comme... katrinleinweber authored on 28 May 2015	e64fd58

```
# Import function for .MWX files created by the
# "Swift II Multi Wavelength Version 2.00" program (Biochrom Ltd)
# Important: Ensure MWX export by activating "Export Spreadsheet" under
# "Run > Medhod > Run Options"

library(reshape2) # needed for colsplit

import_MWX_files <- function(skip_lines,
                             # parameter defaults
                             clean = TRUE,
                             var_sep = FALSE,
                             new_vars = FALSE
) {

  # Please note: MWX files should be present in R's working directory, and
  # should have resulted from same method. Otherwise, different numbers of
  # header lines, variables, etc. may cause this script to run into errors.
  MWX_files <- list.files(getwd(),
                          # setwd(dirname(file.choose())) # alternative selection
                          # of working directory via file selection dialogue
                          pattern = "\\.MWX"
  )

  # merge data tables from MWX file into R dataframe
  MWX_df <- do.call("rbind",
```

Import function for .MWX files created by "Swift II Multi Wavelength Version 2.00"

GitHub, Inc. (US) https://github.com/katrinleinweber/MWX-import/commits/master

Commits on Oct 20, 2016

CMD+I in RStudio to demonstrate GitHub's blame UI katrinleinweber committed 6 minutes ago	5778e71

Commits on May 28, 2015

add Zenodo-DOI katrinleinweber committed on 28 May 2015	4a2f690
improve explanations, comments & object names katrinleinweber committed on 28 May 2015	e64fd58

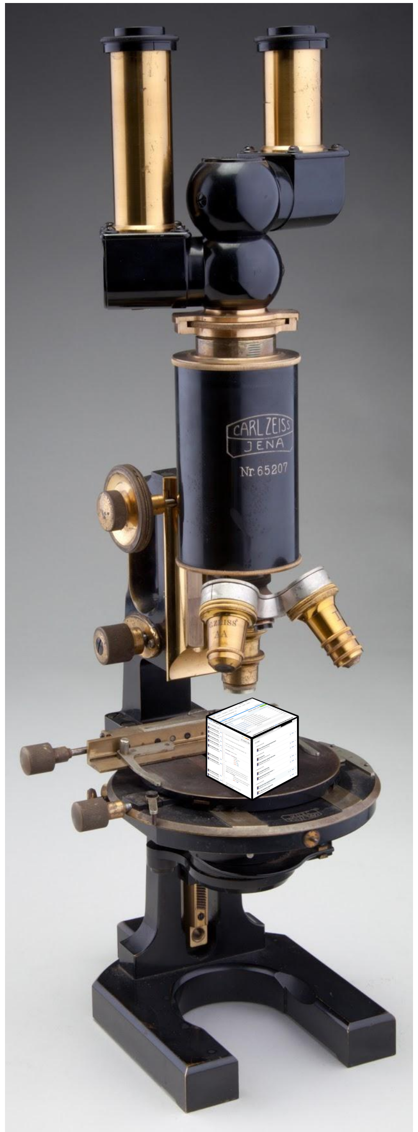
Commits on Apr 17, 2015

fix inconsistent indents & typo katrinleinweber committed on 17 Apr 2015	5745228
boolean improvements thanks to @enplotz ... katrinleinweber committed on 17 Apr 2015	4cb5599

Commits on Apr 16, 2015

initial commit: script, example data & basic docu katrinleinweber committed on 16 Apr 2015	a35bb4f
Added .gitattributes & .gitignore files katrinleinweber committed on 16 Apr 2015	6577de5

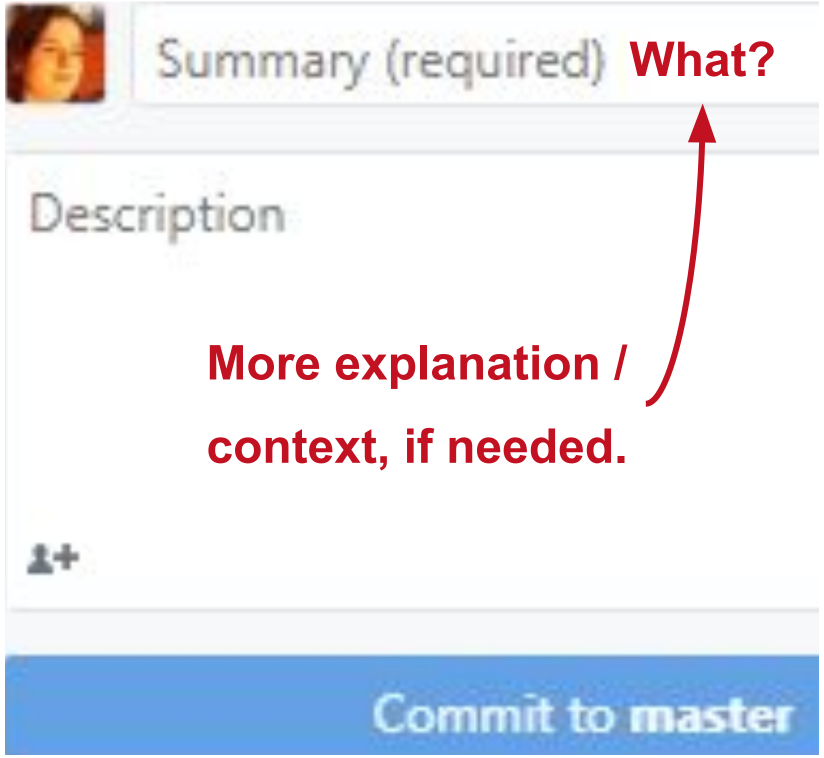
pxhere.com/en/photo/757871
CC0 Public domain



By Chad Anderson, staff photographer for SFO Museum CC BY-SA 2.0
via commons.Wikimedia.org/w/index.php?curid=45625745 (cropped)

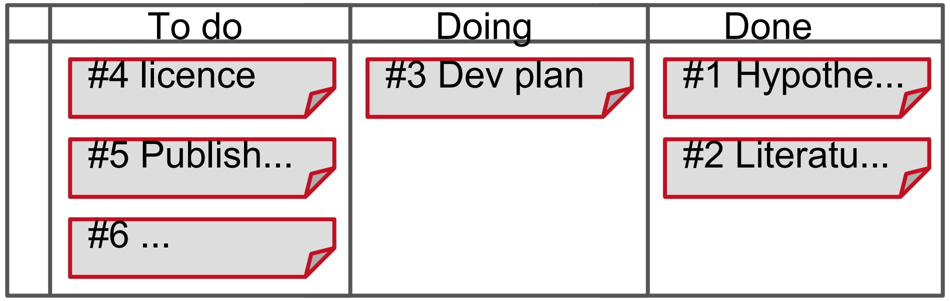
Accessible for your future self, collaborators, students, reviewers

- commit messages should explain answer mostly the “Why?”
  - explanations besides history / `log` & `blame` view
  - demo: [GitHub.com/TIBHannover/BacDiveR/commits]() & [/blame/master/R/aggregate_datasets.R]()
  - pull/merge request titles can answer only the “What?”
- diffs should be small, logical, atomic, also across many files
  - fixing typos & links => `git commit -m “Proof-read”`
  - code & test => `git commit -m “Implement feature ...”`



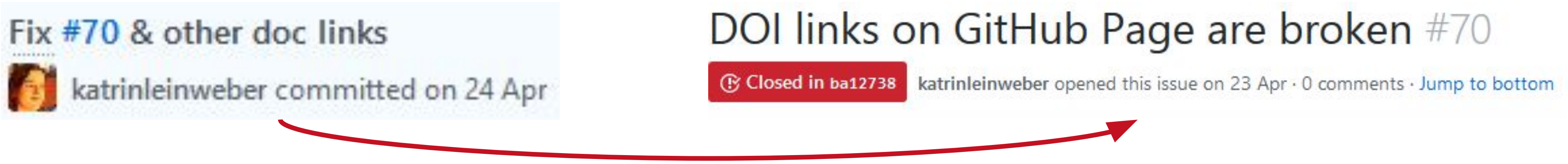
Git is both a communication channel when necessary, and a communication-avoidance system when not.

Issue tracking = project management



To do	Doing	Done
#4 licence	#3 Dev plan	#1 Hypothe...
#5 Publish...		#2 Literatu...
#6 ...		

- issue = idea, discussion, problem report, question, etc.
- `- [ ] task` lists, `@`-mentions, `#`-links to other issues
- labels, assignees, milestones / due dates, etc.
- (peer-)reviewing pull/merge request = knowledge transfer within team
- “closing keywords” on most Git platforms: `fix #...` in commit message



Fix #70 & other doc links

katrinleinweber committed on 24 Apr

DOI links on GitHub Page are broken #70

katrinleinweber opened this issue on 23 Apr · 0 comments · Jump to bottom

issue tracker = “living plan”, community workspace, discussion forum, work sheet, etc.

Version control beyond software source code

Tenen & Wythoff (2014)
ProgrammingHistorian.org/en/lessons/sustainable-authorship-in-plain-text-using-pandoc-and-markdown

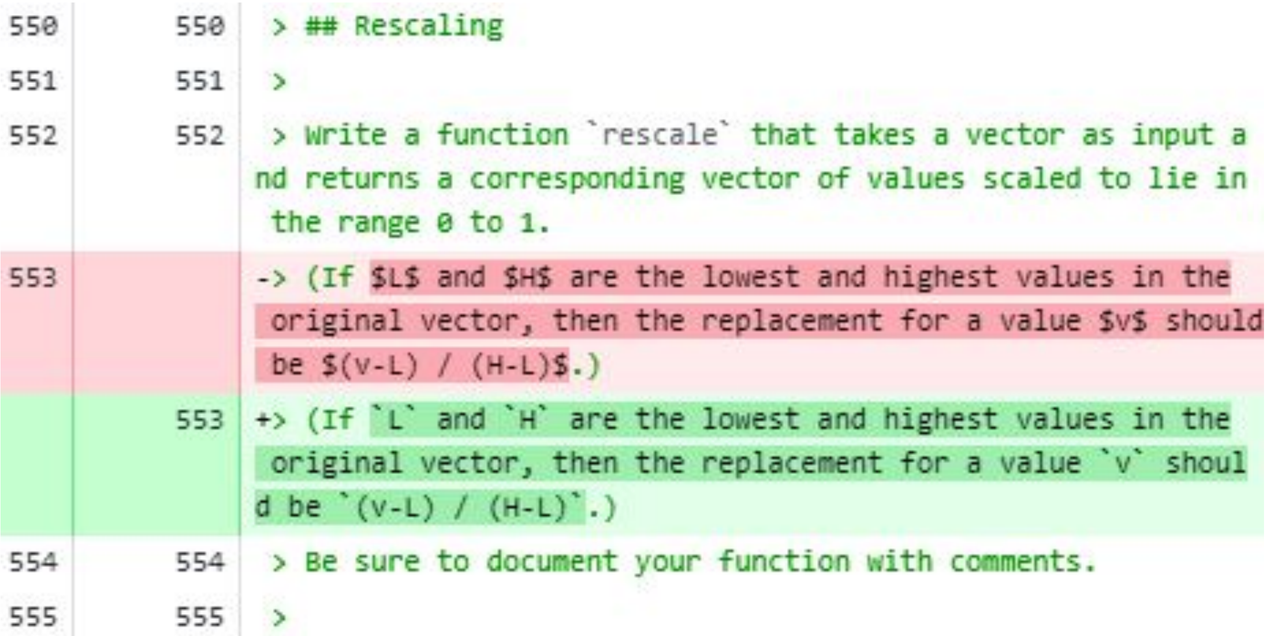
- large, non-text files => Git-LFS.com
- text documents: Markdown, LaTeX, GitHub/Lab Pages
  - alternative to fast-syncing tools like GDoc, EtherPad, HackMD, etc.
  - only problem: keeping diffs small & readable => break lines after N characters
  - up & coming tools: Overleaf.com

    GitBook.com

    Authorea.com

    PenFlip.com

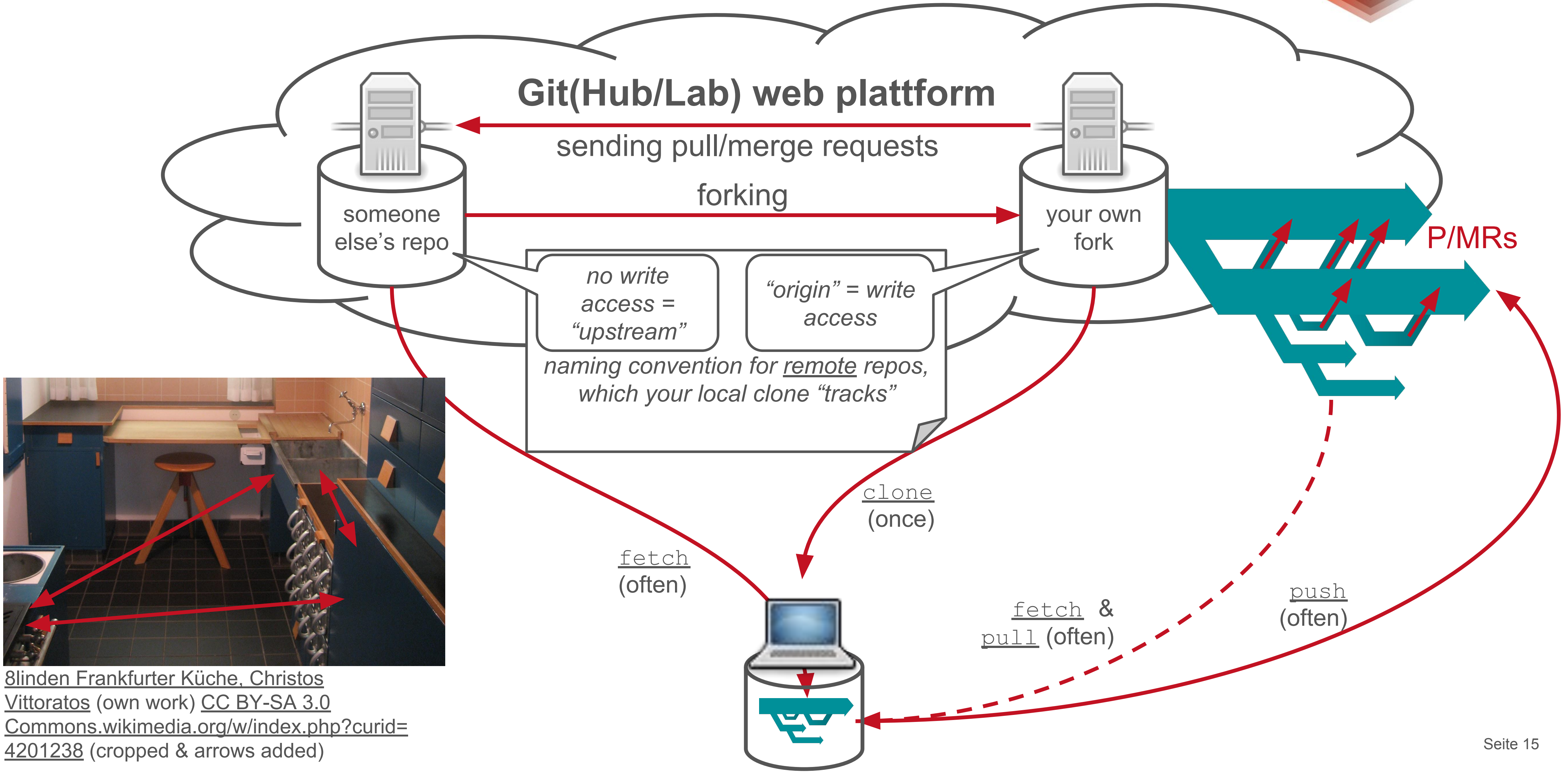
Bundes-Git@ec6a9c3d#diff-5eaf7301...L595



```
> ## Rescaling
>
> Write a function `rescale` that takes a vector as input a
nd returns a corresponding vector of values scaled to lie in
 the range 0 to 1.
-> (If $L$ and $H$ are the lowest and highest values in the
 original vector, then the replacement for a value $v$ should
 be $(v-L) / (H-L)$.)
+> (If `L` and `H` are the lowest and highest values in the
 original vector, then the replacement for a value `v` shoul
d be `(v-L) / (H-L)`.)
> Be sure to document your function with comments.
>
```

Git workflows (correct typos/links in WUI & CLI)

Git workflow



Git(Hub/Lab) web plattform

sending pull/merge requests

forking

someone else's repo

your own fork

P/MRs

no write access = "upstream"

"origin" = write access

naming convention for <u>remote</u> repos, which your local clone "tracks"

`clone` (once)

`fetch` (often)

`fetch` & `pull` (often)

`push` (often)

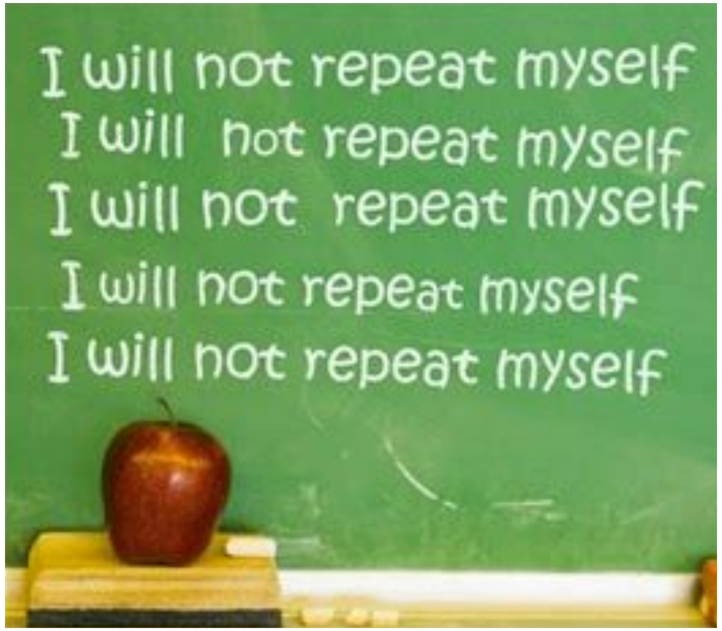
8linden Frankfurter Küche, Christos Vittoratos (own work) CC BY-SA 3.0 Commons.wikimedia.org/w/index.php?curid=4201238 (cropped & arrows added)

Accessibility Agenda

1. Definitions & roles
2. Version control & project management with Git(Hub)
3. Lunch
4. **“Accessible” software ≈ “comprehensible” code**
   - **Functions in Python & R**

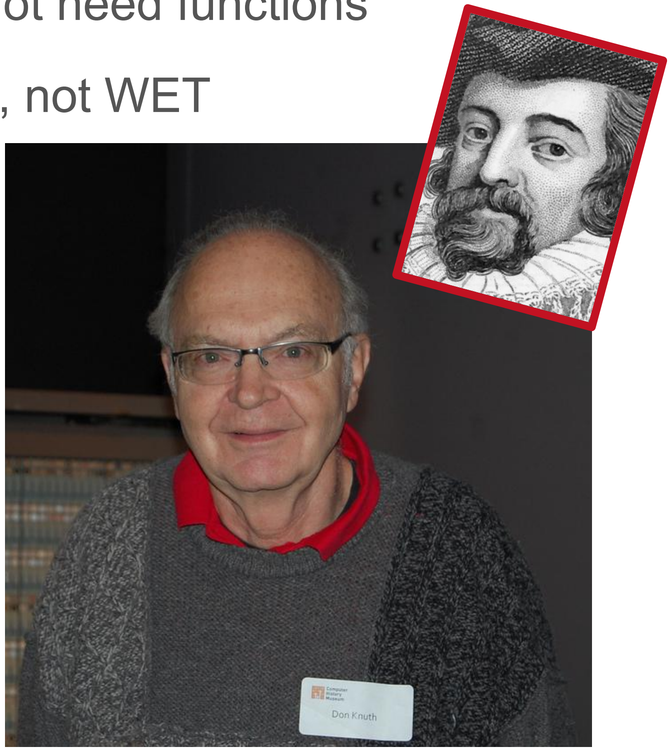
iterative, continuous improvement

Functions ("subroutines" or "procedures")



- 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"
- Python:
  GitHub.com/TIBHannover/2018-07-09-FAIR-Data-and-Software/tree/gh-pages/code/Python &
  KatyHuff.GitHub.io/python-testing

LEIBNIZ INFORMATION CENTRE
FOR SCIENCE AND TECHNOLOGY
UNIVERSITY LIBRARY

TIB

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