Open Science and Artificial Intelligence in Research Software Engineering

Lecturer: Daniel Garijo

ETSI Informáticos, Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
https://oeg.fi.upm.es/

Session 2: Initiatives to describe Research Data and Software*

daniel.garijo@upm.es
@dgarijov

*With slides from "The Scientific Paper of the Future" training materials for the OntoSoft project (Gil et al)

License

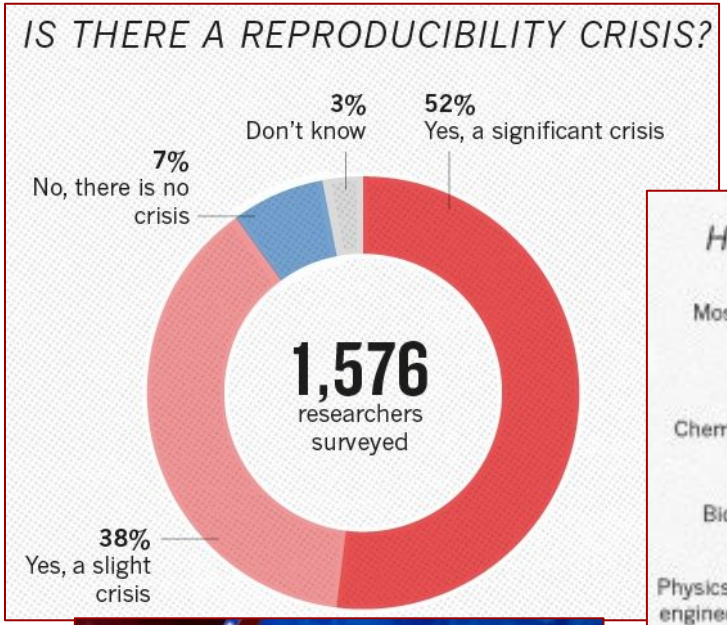
You are free:

- to Share — to copy, distribute and transmit the work
- to Remix — to adapt the work

Under the following conditions

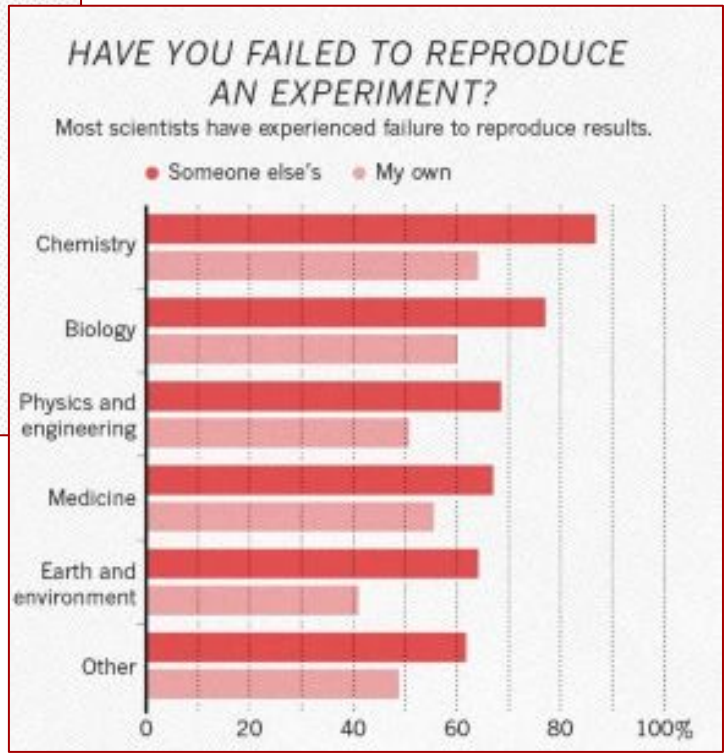
- Non-commercial – You cannot use it for commercial purposes, nor for training inside a commercial company
- Attribution — You must attribute the work by inserting
- “[source http://www.oeg-upm.net/]” at the footer of each reused slide
- A credits slide stating: “These slides are partially based on ‘Open Science and Artificial Intelligence in Research Software Engineering: Introduction’ by Daniel Garijo and Oscar Corcho”
- Share-Alike

Previously on OS and AI in RSE…



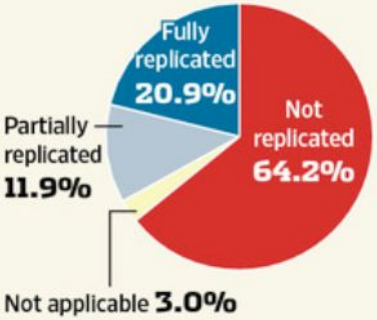
Retraction Watch

Tracking retractions as a window into the scientific process



No Cure

When Bayer tried to replicate results of 67 studies published in academic journals, nearly two-thirds failed.

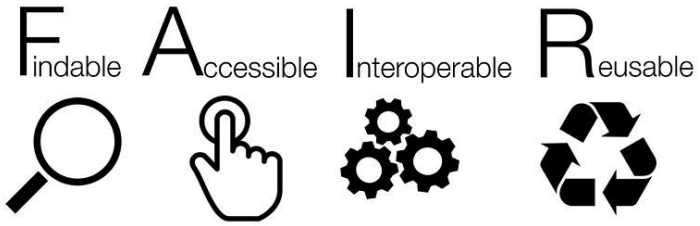


Source: Nature Reviews Drug Discovery

https://www.nature.com/articles/533452a

https://www.force11.org/group/fairgroup/fairprinciples

https://doi.org/10.1038/sdata.2016.18



To be Findable:

F1. (meta)data are assigned a globally unique and eternally persistent identifier.
F2. data are described with rich metadata.
F3. (meta)data are registered or indexed in a searchable resource.
F4. metadata specify the data identifier.

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 are accessible, even when the data are no longer available.

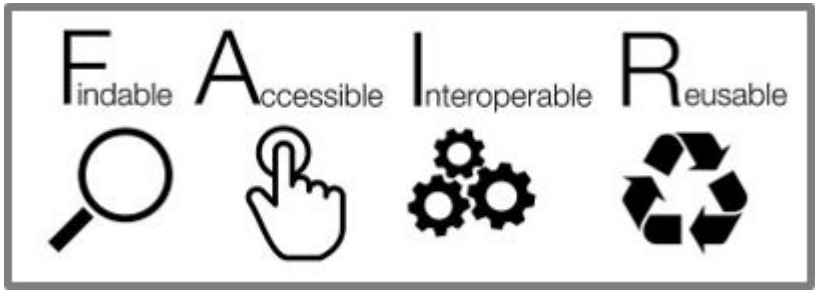
TO BE INTEROPERABLE:

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.

TO BE RE-USABLE:

R1. meta(data) have a plurality of accurate and relevant attributes.
R1.1. (meta)data are released with a clear and accessible data usage license.
R1.2. (meta)data are associated with their provenance.
R1.3. (meta)data meet domain-relevant community standards.

Extensions and alternatives to FAIR



Data (initially) [1]

Research Software (FAIR4RS)

Methods (FAIR Wfs)

Semantic artefacts

Other guidelines:

- Guidelines for Transparency and Openness Promotion (TOP) [2]
- Reproducibility Enhancement Principles (REP) [3]
- ...

RDA
RESEARCH DATA ALLIANCE

FORCE11
The Future of Research Communications and e-Scholarship

[1] Wilkinson, M., Dumontier, M., Aalbersberg, I. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3,** 160018 (2016). https://doi.org/10.1038/sdata.2016.18
[2] https://www.cos.io/initiatives/top-guidelines
[3] Stodden, V et al Enhancing reproducibility for computational methods https://www.science.org/lookup/doi/10.1126/science.aah6168

Best practices for

- Making Research Data accessible and reusable
- Making Research Software accessible and reusable

Using

- Shared repositories
- Persistent identifiers
- Rich metadata descriptions
- Open licenses

Making your data look tasty…

Best practices for making research data accesible and reusable

Image generated with Dall-e (https://labs.openai.com/)

Problems with current practice

Nature Genetics **41**, 149 - 155 (2009)
Published online: 28 January 2008 | doi:10.1038/ng.295

Repeatability of published microarray gene expression analyses

scientists. Here we evaluated the replication of data analyses in 18 articles on microarray-based gene expression profiling published in *Nature Genetics* in 2005–2006. One table or figure from each article was independently evaluated by two teams of analysts. We reproduced two analyses in principle and six partially or with some discrepancies; ten could not be reproduced. The main reason for failure to reproduce was data unavailability, and discrepancies were mostly due to incomplete data annotation or specification of data processing and analysis.

- Data **not made available at all**

PLOS ONE | DOI:10.1371/journal.pone.0115253 December 26, 2014

RESEARCH ARTICLE

Scholarly Context Not Found: One in Five Articles Suffers from Reference Rot

Martin Klein[1]*, Herbert Van de Sompel[1], Robert Sanderson[1], Harihar Shankar[1], Lyudmila Balakireva[1], Ke Zhou[2], Richard Tobin[2]

We analyze a vast collection of articles from three corpora that span publication years 1997 to 2012. For over one million references to web resources extracted from over 3.5 million articles, we observe that the fraction of articles containing references to web resources is growing steadily over time. We find **one out of five STM articles suffering from reference rot**, meaning it is impossible to revisit the web context that surrounds them some time after their publication. When only considering STM articles that contain references to web resources, this fraction increases to seven out of ten.

- Data made available, but **URL does not resolve**

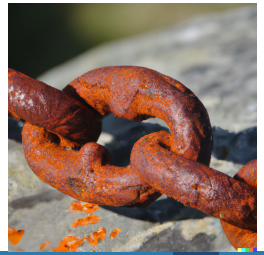


Image generated with Dall-e (https://labs.openai.com/)

Data papers

Ecological Research
July 2013, Volume 28, Issue 4, p 541

Date: 10 May 2013

Monitoring records of plant species in the Hakone region of Fuji-Hakone-Izu National Park, Japan, 2001–2010

Takeshi Osawa

Abstract

The monitoring of species occurrences is a crucial aspect of biodiversity conservation, and regional volunteerism can serve as a powerful tool in such endeavors. The Fuji-Hakone-Izu National Park in the Hakone region of Kanagawa Prefecture, Japan, boasts a volunteer association of approximately 100 members. These volunteers have monitored plant species occurrences from 2001 to the present along several hiking trails in the region. In this paper, I present the annual observation records of plant occurrences in Hakone from 2001 to 2010. This data set includes 1,071 species of plants from 151 families. Scientific names follow the Y List, and this data set includes several threatened plant species. Data files are formatted based on the Darwin Core and Darwin Core Archives, which are defined by the Biodiversity Information Standards (BIS) or Biodiversity Information Standards Taxonomic Databases Working Group (TDWG). Data files filled on required and some additional item on Darwin Core. The data set can download from the author's personal Web site as of July 2012. These data will soon be published for the Global Biodiversity Information Facility (GBIF) through GBIF Japan. All users can then access the data from the GBIF portal site.

- *The complete data set for this abstract published in the Data Paper section of the journal is available in electronic format in Ecological Research Data Paper Archives at http://db.cger.nies.go.jp/JaLTER/ER_DataPapers/archives/2013/ERDP-2013-01.*

Domain-specific repositories

LTER The US Long Term Ecological Research Network

+/- NTL LTER *"WDNR Yahara Lakes Fisheries: Fish Lengths and Weights 1987-1998" - Lathrop*

LTER Identifier:

knb-lter-ntl.279.1

Abstract:

These data were collected by the Wisconsin Department of Natural Resources (WDNR) from 1987-1998. Most of these data (1987-1993) precede 1995, the year that the University of Wisconsin Â NTL-LTER program Â took over sampling of the Yahara Lakes. However, WDNR data collected from 1997-1998 Â (unrelated to LTER sampling) is also included. In 1987 a joint project by the WDNR and the University of Wisconsin-Madison, Center for Limnology (CFL) was initiated on Lake Mendota. The project involved biomanipulation o...

Owners/Creators:

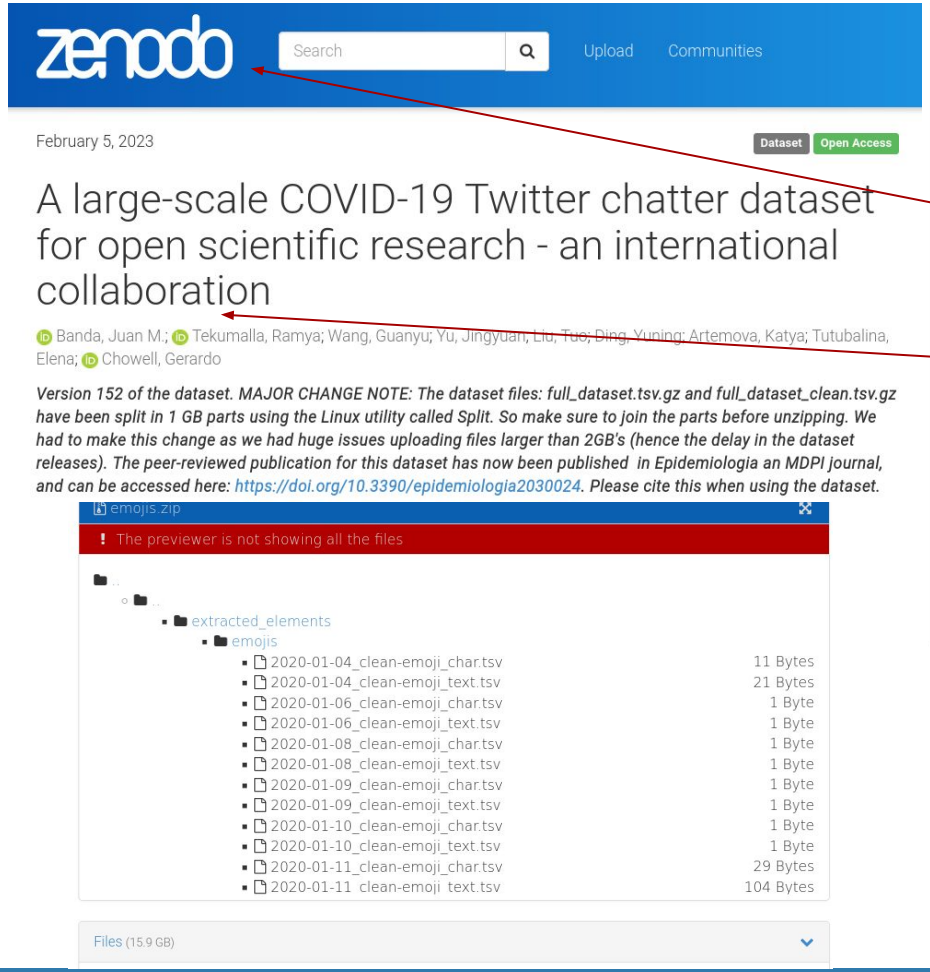
Lathrop

Metadata:

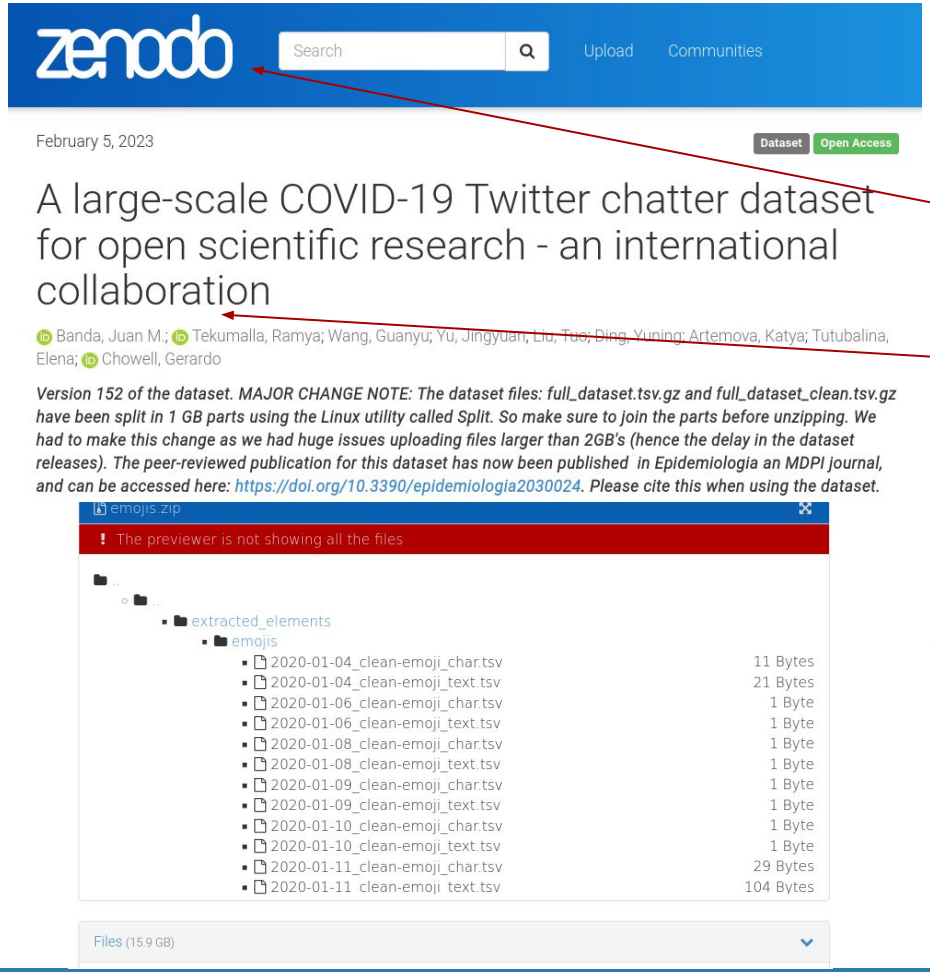
Select here for full metadata

Data File(s):

- wdnr_fyke_minifyke_seine_lengths_weights.csv
- wdnr_boomshock_lengths_weights.csv
- wdnr_gillnet_lengths_weights_93.csv
- wdnr_walleye_age_lengths_weights_87.csv
- wdnr_creel_survey_lengths_weights.csv
- wdnr_creel_survey_angler_counts.csv



1. Publish in a shared repository
2. Add minimal metadata
3. Add domain-specific metadata
4. Use unique persistent identifiers
5. Declare citation preference



1. **Publish in a shared repository**
2. Add minimal metadata
3. Add domain-specific metadata
4. Use unique persistent identifiers
5. Declare citation preference

Shedding Light on the Dark Data in the Long Tail of Science

P. Bryan Heidorn

From: Library Trends
Volume 57, Number 2, Fall 2008
pp. 280-299 | 10.1353/lib.0.0036

Abstract:

One of the primary outputs of the scientific enterprise is data, but many institutions such as libraries that are charged with preserving and disseminating scholarly output have largely ignored this form of documentation of scholarly activity. This paper focuses on a particularly troublesome class of data, termed *dark data*. "Dark data" is not carefully indexed and stored so it becomes nearly invisible to scientists and other potential users and therefore is more likely to remain underutilized and eventually lost. The article discusses how the concepts from long-tail economics can be used to understand potential solutions for better curation of this data. The paper describes why this data is critical to scientific progress, some of the properties of this data, as well as some social and technical barriers to proper management of this class of data. Many potentially useful institutional, social, and technical solutions are under development and are introduced in the last sections of the paper, but these solutions are largely unproven and require additional research and development.

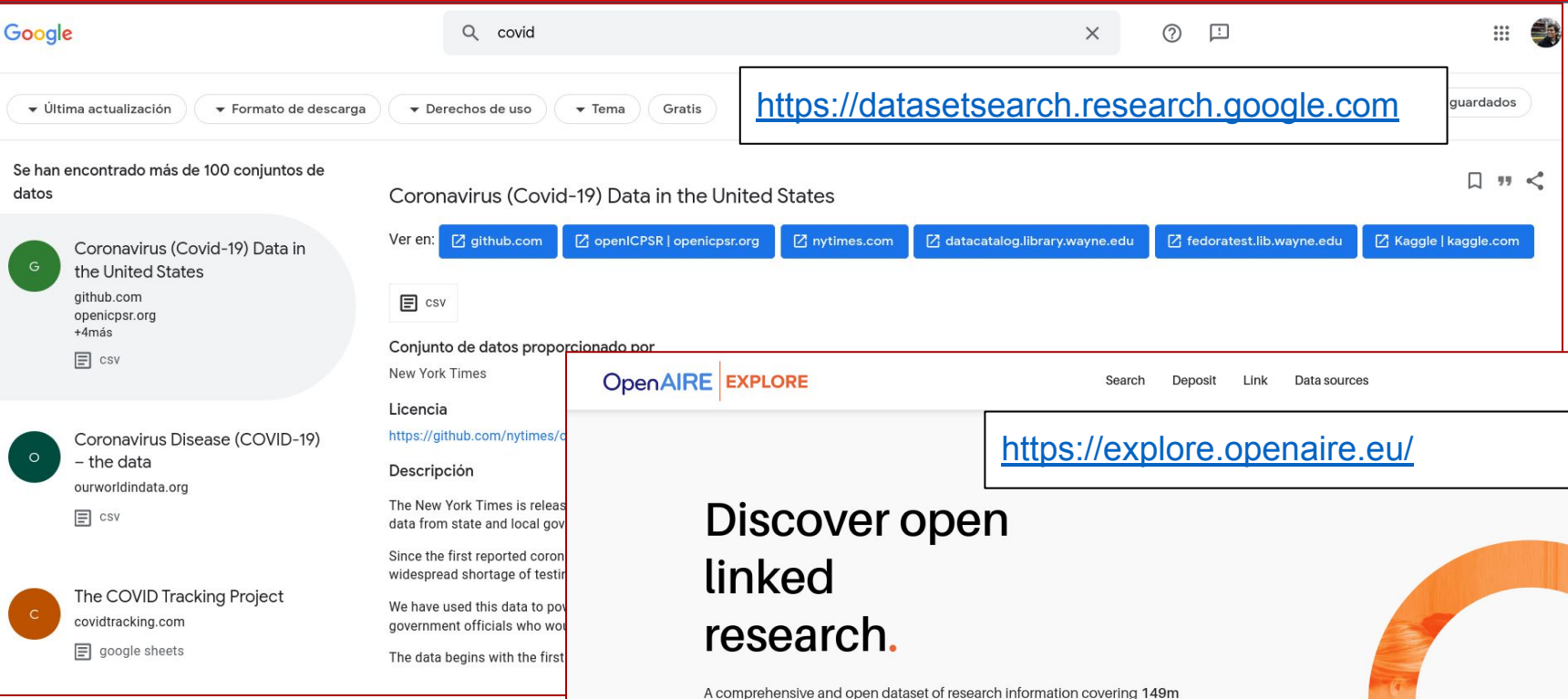
Not curated

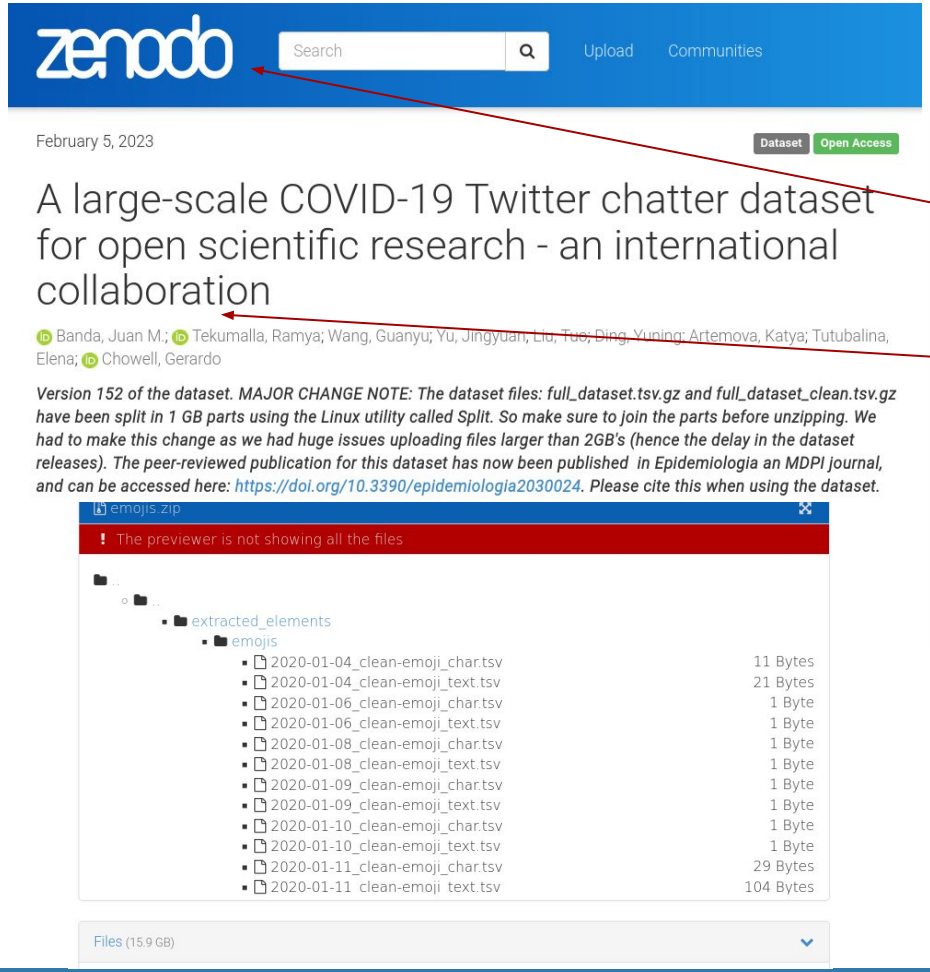
Curated

"Pangaea logo hg" by Hannes Grobe/AWI - Own work. Licensed under CC BY 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Pangaea_logo_hg.png#mediaviewer/File:Pangaea_logo_hg.png

http://www.arqhys.com/articulos/ingeniero-inspector.html

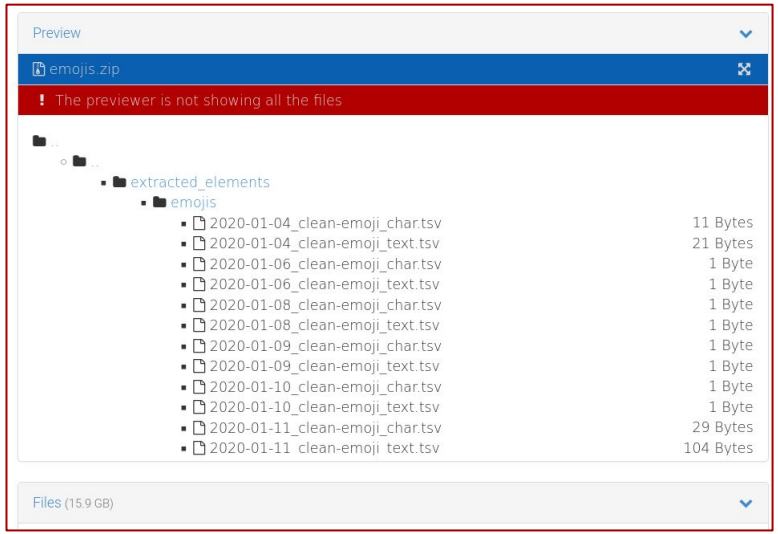
Aggregators of datasets

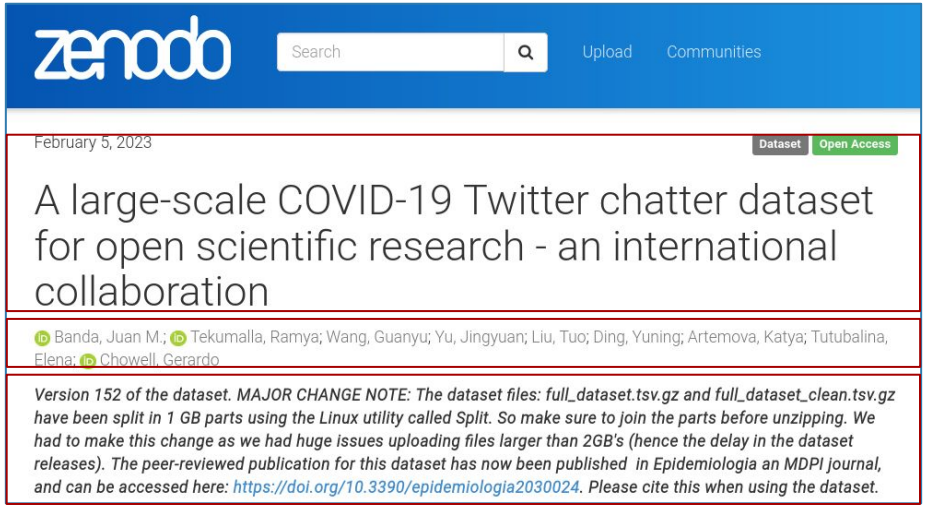




1. Publish in a shared repository
2. **Add minimal metadata**
3. Add domain-specific metadata
4. Use unique persistent identifiers
5. Declare citation preference

Data versus metadata





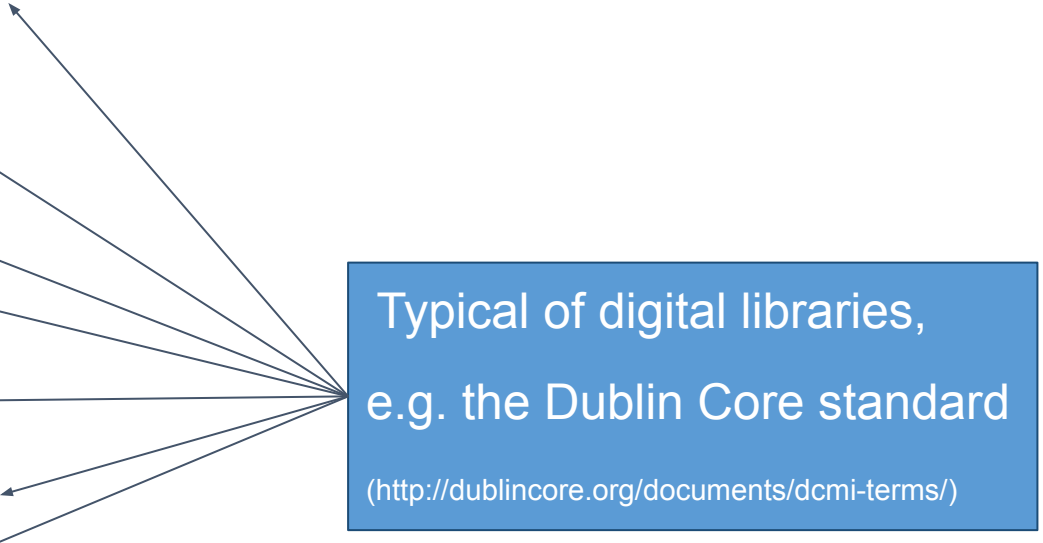
Data: the actual files (tsv, zip, txt, csv, etc)

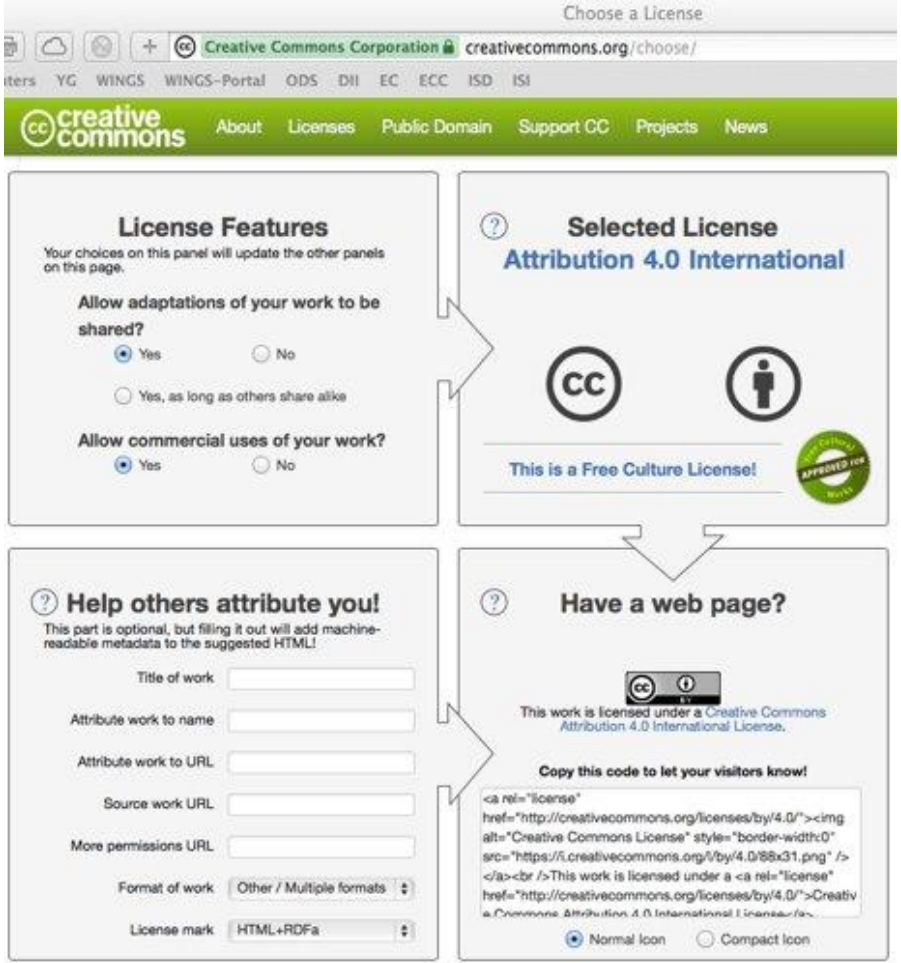
Metadata: fields describing the data (title, authors, description, license, keywords…)

Exercise: what do I get when I resolve the identifier of a dataset in Zenodo?

Minimal metadata

- Dataset name/title
- Description
- Creator(s)
- Publication date
- License
- Publisher/contact
- Version
- Resource type
- Location of the data





Recommended: CC-BY and CC0

CC0 (datasets)

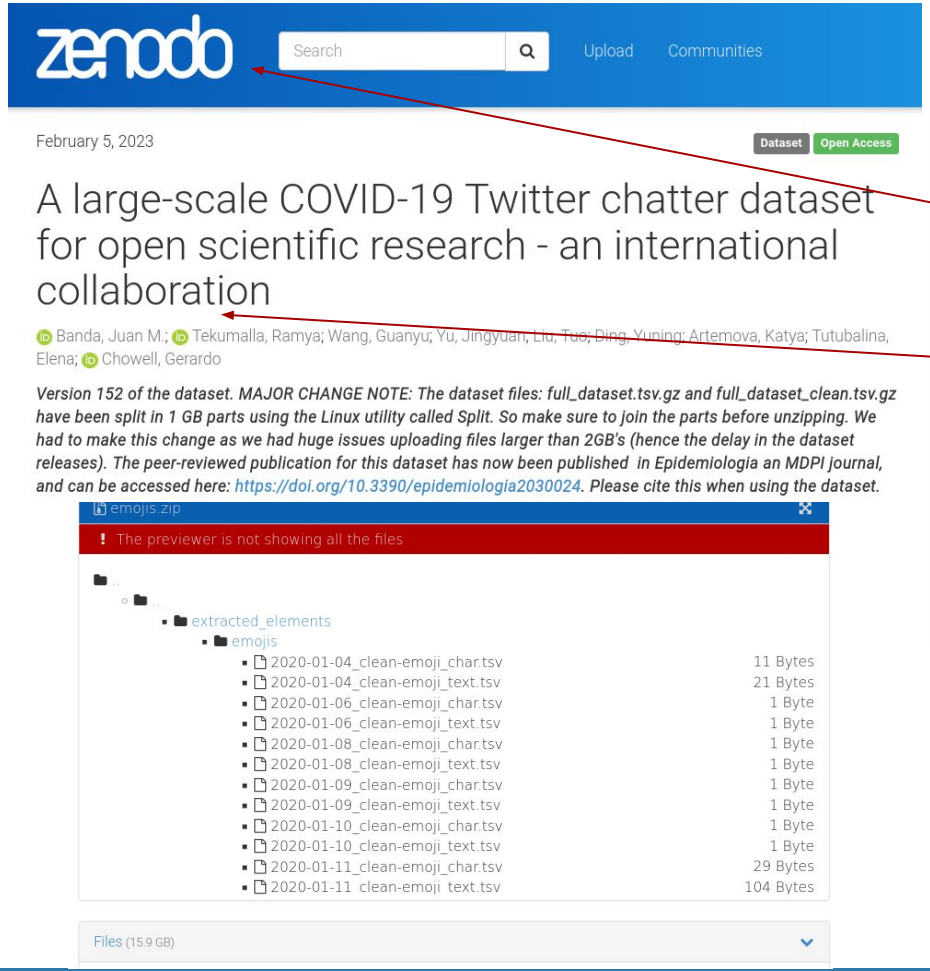
CC0 can be particularly important for the sharing of data and databases, since it otherwise may be unclear whether highly factual data and databases are restricted by copyright or other rights. Databases may contain facts that, in and of themselves, are not protected by copyright law.

CC0 is recommended for data and databases and is used by hundreds of organizations. It is especially recommended for scientific data. Although CC0 doesn't legally require users of the data to cite the source, it does not take away the moral responsibility to give attribution, as is common in scientific research.

Attribution
CC BY

This license lets others distribute, remix, tweak, and build upon your work, even commercially, as long as they credit you for the original creation. This is the most accommodating of licenses offered. Recommended for maximum dissemination and use of licensed materials.

http://creativecommons.org/licenses/



1. Publish in a shared repository
2. Add minimal metadata
3. **Add domain-specific metadata**
4. Use unique persistent identifiers
5. Declar citation preference

Domain-specific metadata

General

- Dataset name/title
- Description
- Creator(s)
- Publication date
- License
- Publisher/contact
- Version
- Resource type
- Location of the data

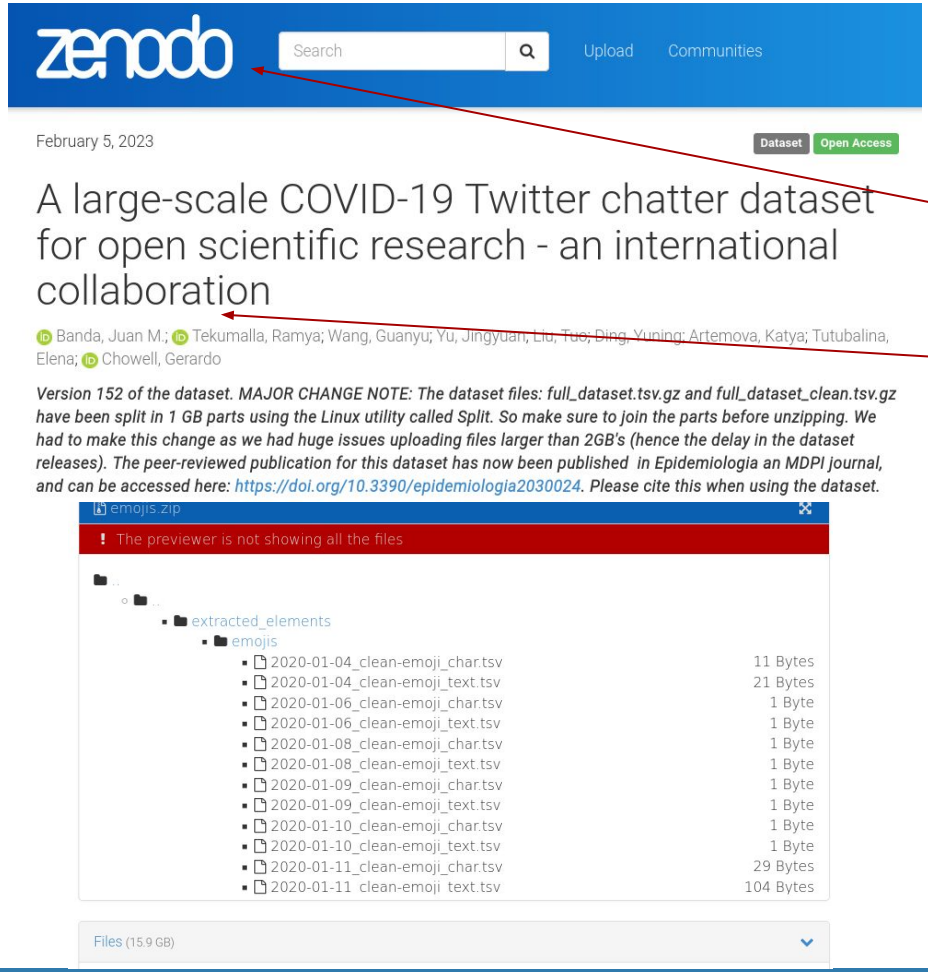
Specific

- Domain keywords
- Preprocessing
- Dataset Features (variables)
- Visualization

Domain standards are used to describe datasets in their domain(s)

Global climate observing system

Overview of best practices



1. Publish in a shared repository
2. Add minimal metadata
3. Add domain-specific metadata
4. **Use unique persistent identifiers**
5. Declare citation preference

Main types of unique identifiers

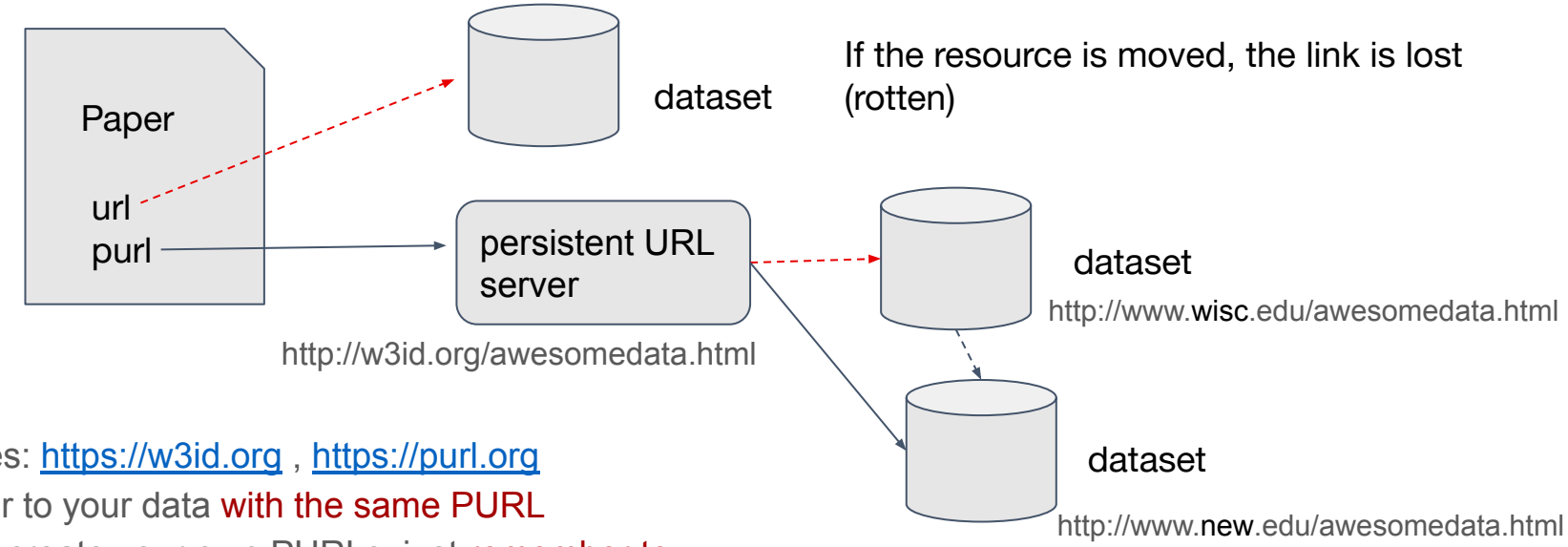
1. Uniform Resource Locator (URL)
2. Persistent URL (purl)
3. Digital Object Identifier (DOI)

Image generated with Dall-e (https://labs.openai.com/)

- Minimal effort to create
- No guarantee of persistence
  a. i.e., almost guaranteed it will not have persistence
  b. e.g., http://www.greatuniversity.edu/students/joesmith/awesomedata
- Not recommended for papers/reports!

Image generated with Dall-e (https://labs.openai.com/)

Persistent proxy URL that can be redirected to your resource:



- Purl services: https://w3id.org , https://purl.org
- Always refer to your data with the same PURL
- It is easy to create your own PURLs, just remember to update it whenever you move the data

PLoS Biol. 2003 Nov; 1(2): e57.
Published online 2003 Nov 17. doi: 10.1371/journal.pbio.0000057
PMCID: PMC261894

The What and Whys of DOIs

Susanne DeRisi, Rebecca Kennison, and Nick Twyman

DOIs can only be issued by a DOI authority (eg a journal publisher) that guarantees to always resolve it

Data repositories can issue DOIs for data

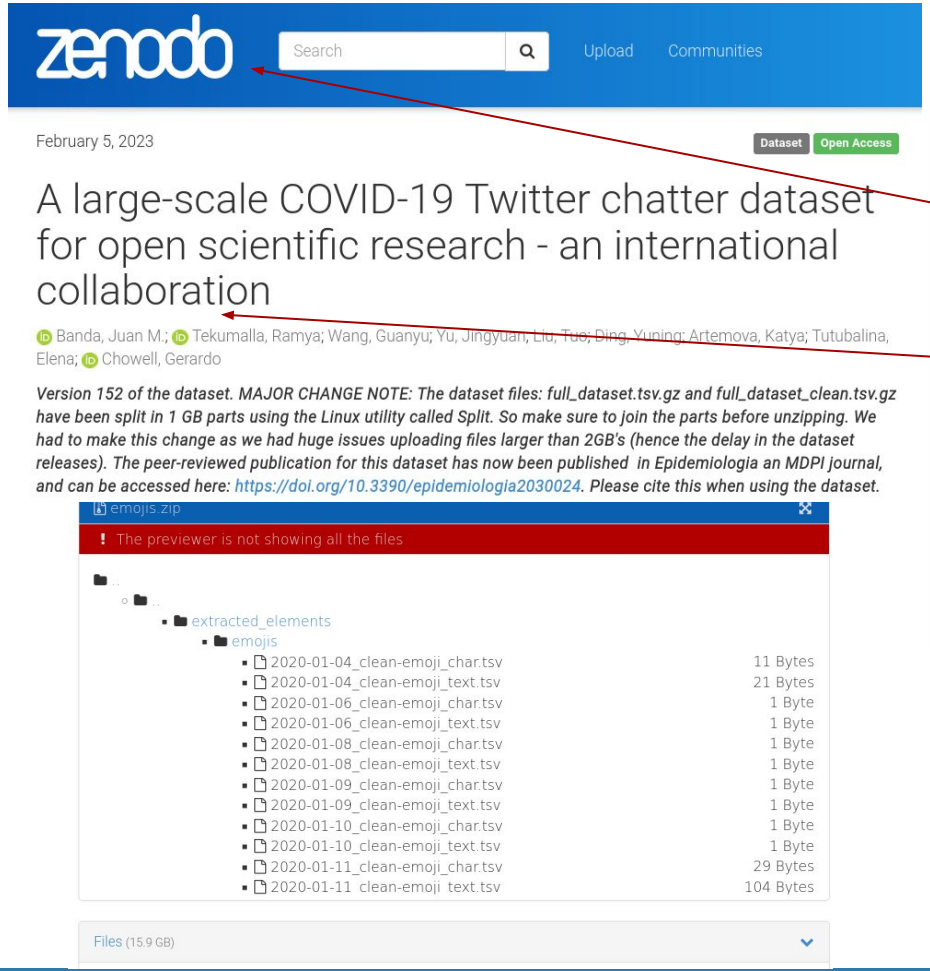
DOIs are **free**

As you may have noticed in the first issue of *PLoS Biology* and again in this issue, there are many pl
where an alphanumeric string appears after the letters "DOI," such as 10.1371/journal.pbio.0000005 or
10.1371/journal.pbio.0000005.g005. Although some of you may already be acquainted with DOIs, others of you may wonder what they are, how they are used, and why we are using them.

What Are DOIs?

Go to:

A Digital Object Identifier (DOI) is an URN (Uniform Resource Name), a compact string that provides a unique, persistent, and actionable identifier for the digital object with which it is associated. DOIs are commonly assigned to scientific articles in their electronic form, but DOIs may also be used as identifiers for any object in any location, although this usage is not yet common outside the online world. The International DOI Foundation (IDF), which governs the DOI system, has several hundred registrant organizations and in August 2003 reported that over 10 million DOIs have been issued since the foundation was created in 1998 (http://www.doi.org/news/03augnews.html).



1. Publish in a shared repository
2. Add minimal metadata
3. Add domain-specific metadata
4. Use unique persistent identifiers
5. **Declare citation preference**

OPEN ACCESS Freely available online

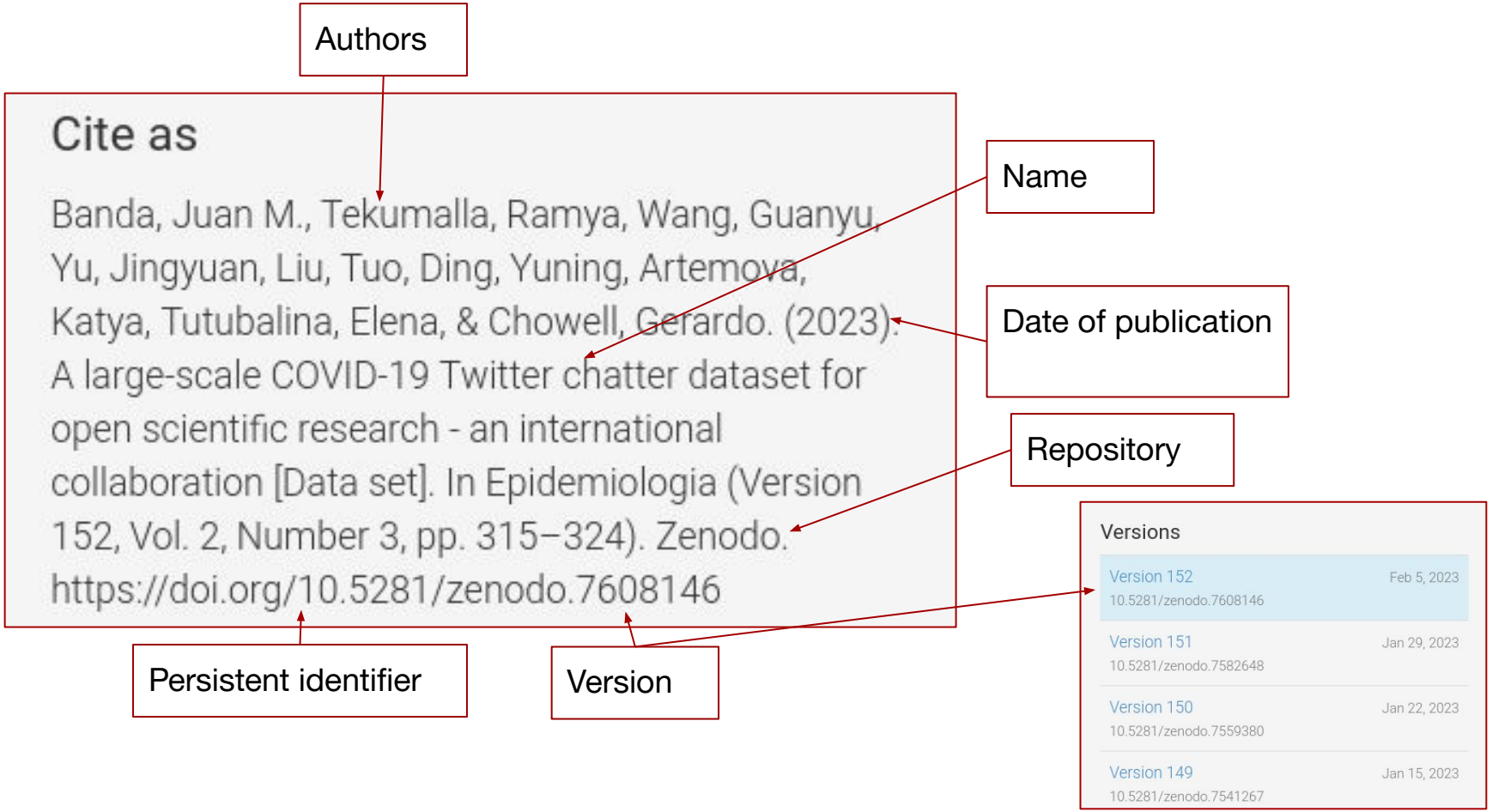
PLoS one

Sharing Detailed Research Data Is Associated with Increased Citation Rate

Heather A. Piwowar*, Roger S. Day, Douglas B. Fridsma

Department of Biomedical Informatics, University of Pittsburgh School of Medicine, Pittsburgh, Pennsylvania, United States of America

Background. Sharing research data provides benefit to the general scientific community, but the benefit is less obvious for the investigator who makes his or her data available. ***Principal Findings.*** We examined the citation history of 85 cancer microarray clinical trial publications with respect to the availability of their data. The 48% of trials with publicly available microarray data received 85% of the aggregate citations. Publicly available data was significantly ($p = 0.006$) associated with a 69% increase in citations, independently of journal impact factor, date of publication, and author country of origin using linear regression. ***Significance.*** This correlation between publicly available data and increased literature impact may further motivate investigators to share their detailed research data.


Authors
Cite as
Banda, Juan M., Tekumalla, Ramya, Wang, Guanyu, Yu, Jingyuan, Liu, Tuo, Ding, Yuning, Artemova, Katya, Tutubalina, Elena, & Chowell, Gerardo. (2023). A large-scale COVID-19 Twitter chatter dataset for open scientific research - an international collaboration [Data set]. In Epidemiologia (Version 152, Vol. 2, Number 3, pp. 315–324). Zenodo. https://doi.org/10.5281/zenodo.7608146
Name
Date of publication
Repository
Versions
Version 152
Feb 5, 2023
10.5281/zenodo.7608146
Version 151
Jan 29, 2023
10.5281/zenodo.7582648
Version 150
Jan 22, 2023
10.5281/zenodo.7559380
Version 149
Jan 15, 2023
10.5281/zenodo.7541267
Persistent identifier
Version

What if…

… there are several datasets in several files?

- Create a DOI for each file and a DOI for the whole set

… the data is from a public repository?

- Publish the query, create a DOI + metadata for it, mention the original source in the metadata, point to the original data source

… the data is from a colleague?

- Get permission in advance and make an agreement, then do as with the data from a public repository

… the data comes from many sources?

- Credit each source, create URIs as needed
- Can create a table with “microattributions” that summarize each data source

… the data comes from a database?

- Create a file (or files) from it

… the data has many versions?

- Create a DOI either for each slice or for each snapshot

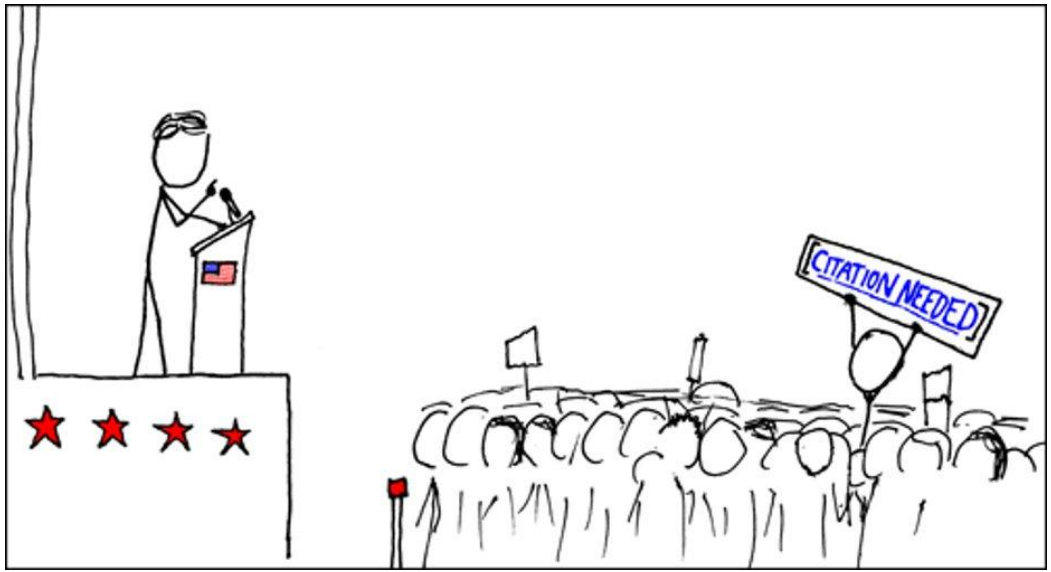
Summary: Simplest approach to make data accessible

1. **Create a public entry for your dataset with a persistent unique identifier**
   - Go to a domain repository (use a general repository, e.g., zenodo.org, if you cannot find one), create an account
   - Create an entry for your dataset
2. **Specify the metadata**
   - Including license -- choose from http://www.creativecommons.org/licenses
3. **Upload/point to the data**

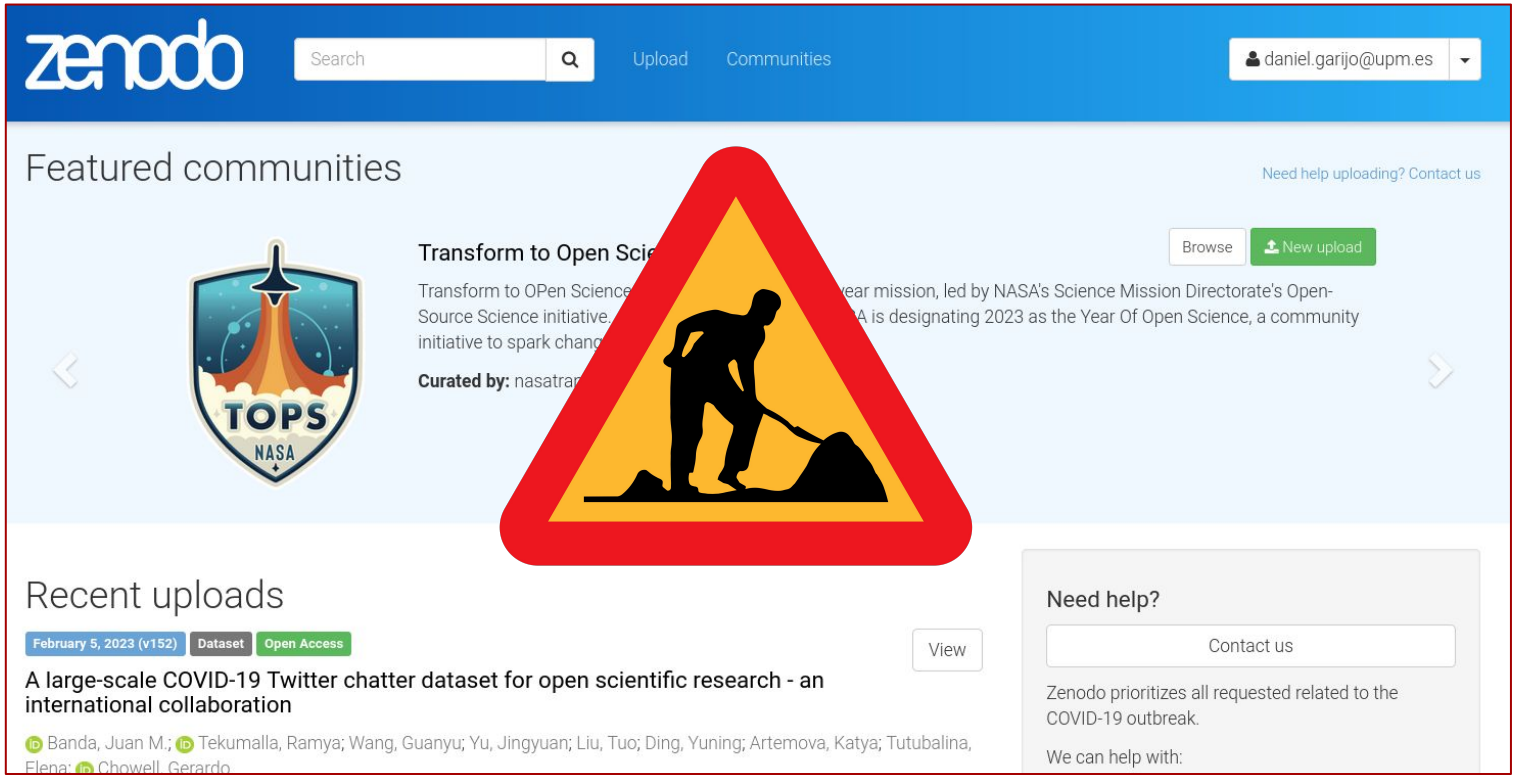
And that 's it! The repository will give you a data citation

Citation goes in the “References” section

- **How to cite the data? You choose:**
- With an in-text pointer as you would cite any other paper (recommended)
- With an in-text pointer in a special “Data Resources” section
- With an in-text pointer in the “Acknowledgments” section



Becoming familiar with Zenodo.org



Making your tools look shiny…

Best practices for making Research **Software accesible and reusable**

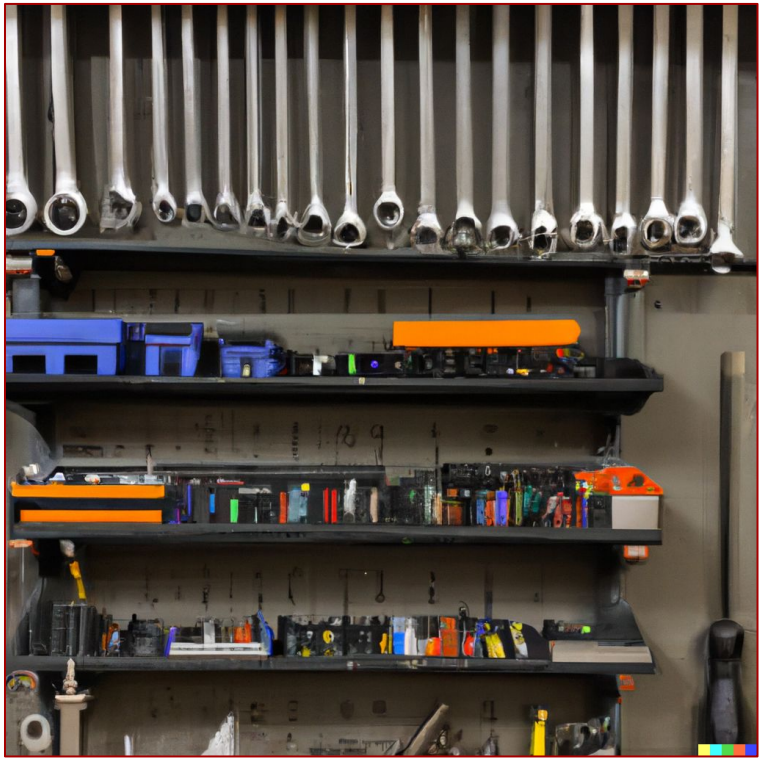
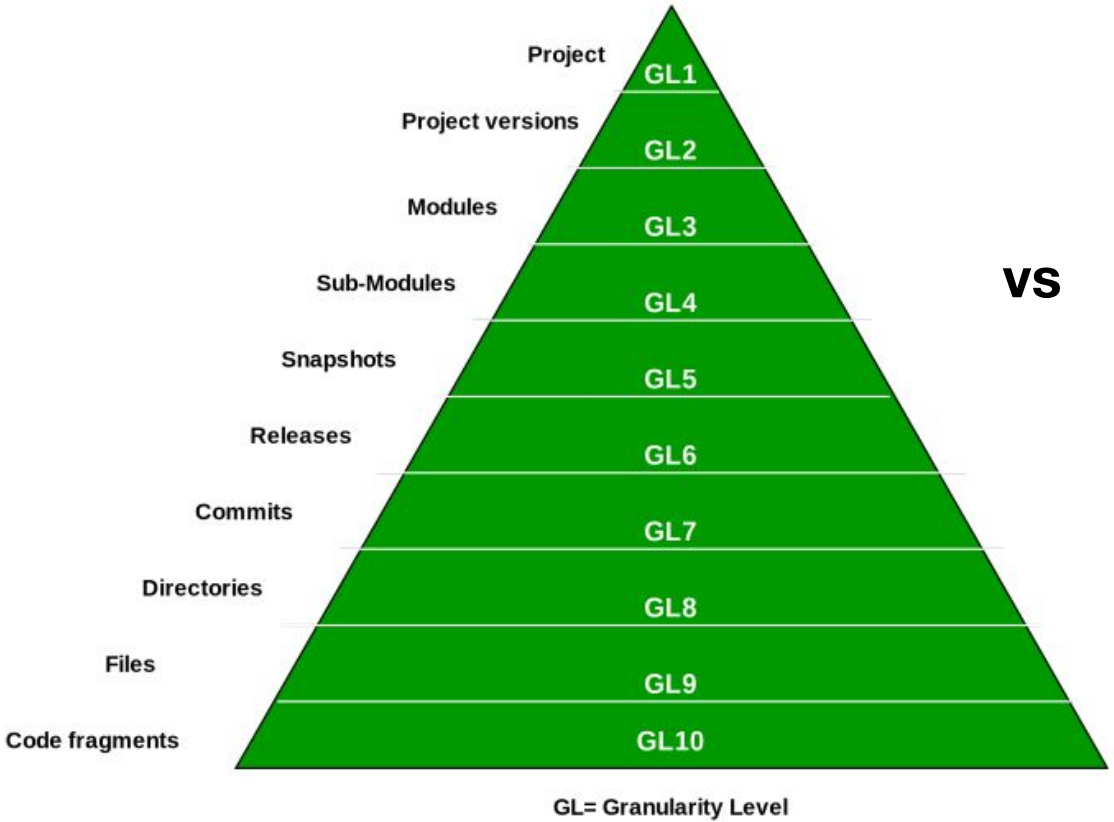


Image generated with Dall-e (https://labs.openai.com/)

Research Software includes source code files, algorithms, scripts, computational workflows and executables that were created during the research process or for a research purpose. Software components (e.g., operating systems, libraries, dependencies, packages, scripts, etc.) that are used for research but were not created during or with a clear research intent should be considered software in research and not Research Software. This differentiation may vary between disciplines.

Article: Chue Hong, Neil P., Katz, Daniel S., Barker, Michelle, Lamprecht, Anna-Lena, Martinez, Carlos, Psomopoulos, Fotis E., Harrow, Jen, Castro, Leyla Jael, Gruenpeter, Morane, Martinez, Paula Andrea, Honeyman, Tom, Struck, Alexander, Lee, Allen, Loewe, Axel, van Werkhoven, Ben, Jones, Catherine, Garijo, Daniel, Plomp, Esther, Genova, Francoise, … RDA FAIR4RS WG. (2022). FAIR Principles for Research Software (FAIR4RS Principles) (1.0). https://doi.org/10.15497/RDA00068

The many layers of Research Software



From https://doi.org/10.15497/RDA00068

vs

Three layers of research software

Analysis code (scripts, workflows, etc.)
Research Prototypes (frameworks)
Research Infrastructure (well established)

From https://docs.google.com/presentation/d/1uwxSwd8chbG7bVn5lPvNNhv5f_JeeelA2mN9NXhlOhA/edit#slide=id.g167ce777f62_0_394

Availability of Software

PLOS supports the development of open source software and believes that, for submissions appropriate open source standards will ensure that the submission conforms to (1) our requi another researcher can reproduce the experiments described, (2) our aim to promote openn PLOS journals can be built upon by future researchers. Therefore, if new software or a new a that the software conforms to the Open Source Definition, have deposited the following three submission as Supporting Information:

- **The associated source code of the software described by the paper.** This should be licensed under a suitable license such as BSD, LGPL, or MIT (see http://www.ope commercial software such as Mathematica and MATLAB does not preclude a paper f preferred.
- **Documentation for running and installing the software.** For end-user applications prerequisite; for software libraries, instructions for using the application program inter
- **A test dataset with associated control parameter settings.** Where feasible, result test data should not have any dependencies — for example, a database dump.

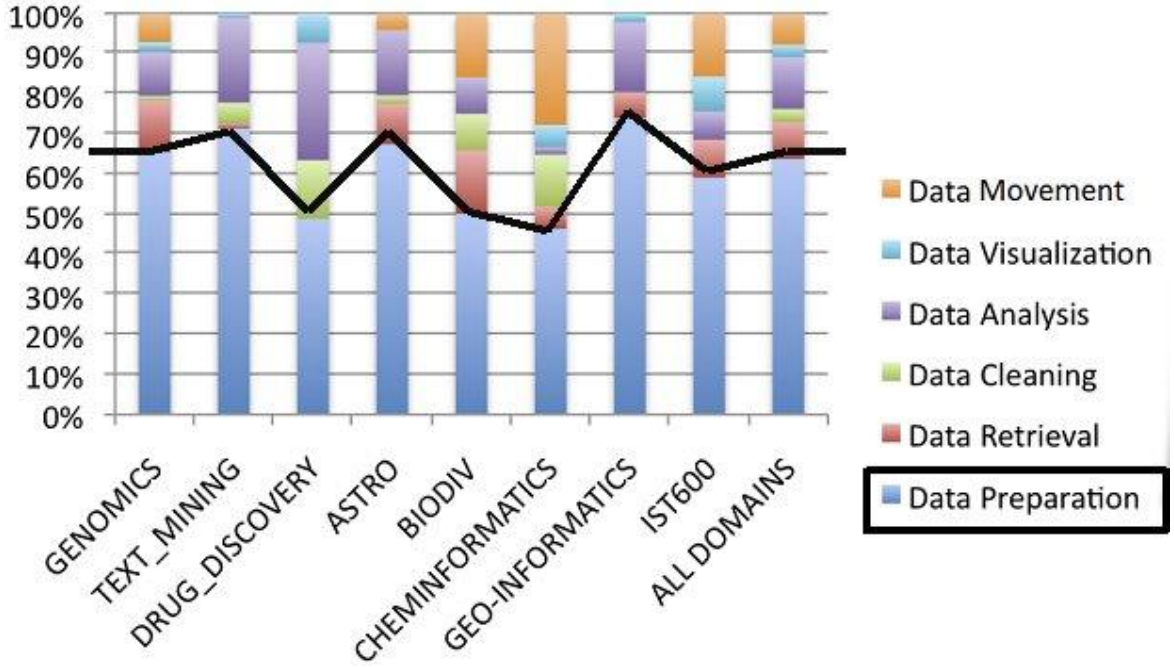
Acceptable archives should provide a public repository of the described software. The code s for creating user accounts, logging in or otherwise registering personal details. The repositor more than 1,000 projects. Examples of such archives are: SourceForge, Bioinformatics.Org, Savannah, GitHub and the Codehaus. Authors should provide a direct link to the deposited s

Why is software not shared?

- “No one would use my code if I shared it”
- “My code is really bad”
- “My code is not ready to be shared”
- “Sharing my software will take a lot of time”
- “I won’t get anything out of sharing my software”
- “I’ve shared software before, bad things happened”
- “I work for the government”
- “I want to commercialize my software”
- “I don’t want anyone to commercialize my software”
- “I don’t know where to start!”

Data preparation software dominates but it's least used

"Scientists and engineers spend more than 60% of their time just preparing the data for model input or data-model comparison" (NASA A40)



"Common Motifs in Scientific Workflows: An Empirical Analysis." Garijo, D.; Alper, P.; Belhajjame, K.; Corcho, O.; Gil, Y.; and Goble, C. Future Generation Computer Systems, 2013.

Software reusability can be challenging

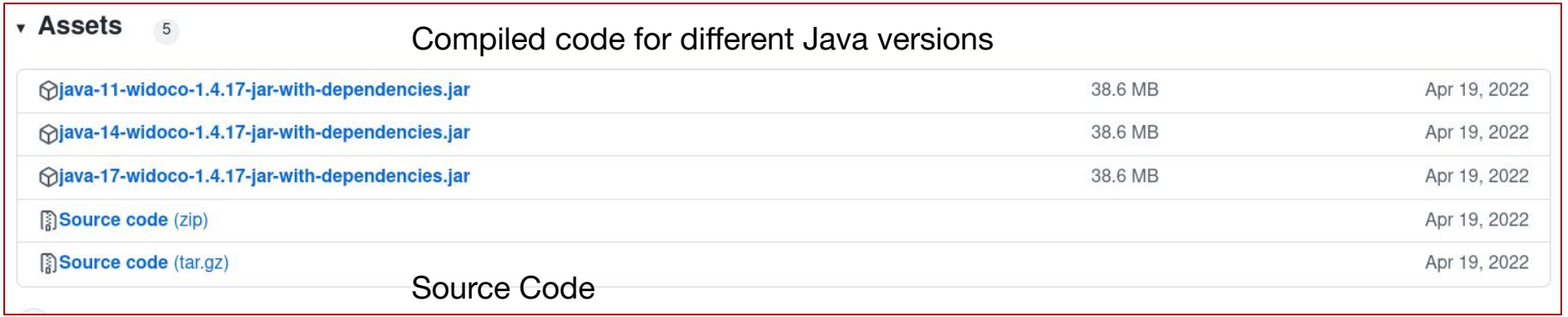
Thoughts on making your software tool ready for publication

1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

Example: Java-based projects

https://github.com/dgarijo/Widoco/releases/tag/v1.4.17



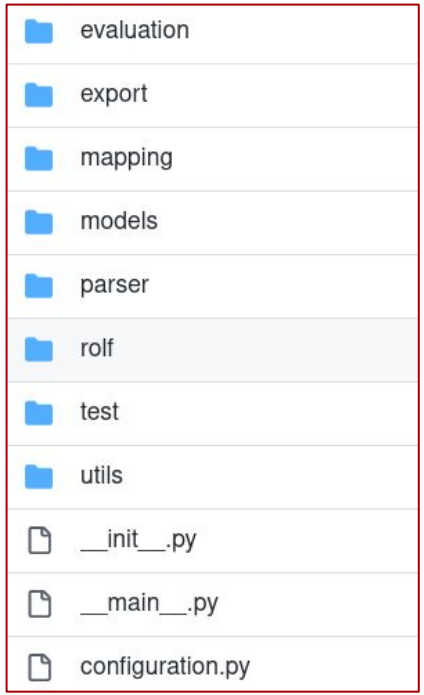
1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

- Document your **dependencies**
- Follow **best practices** in your community to package software artifacts (e.g., setup.py in Python, pom.xml in Java, etc.)
- Do not hardcode file **paths**
- Do not hardcode **inputs**
- Do not hardcode **outputs**
- **Document** your code!
- Rely on **existing** OS libraries
- Use **community standards** for coding and naming

1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

- Separate code in different classes
  - According to their main functionality
  - Separate different classes in different files

- Avoid duplicating code



1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

Do not hardcode **configuration** options

- Use command line commands with options
- Use configuration files

```
$ somef describe --help
  SOMEF Command Line Interface
Usage: somef describe [OPTIONS]

  Running the Command Line Interface

Options:
  -t, --threshold FLOAT           Threshold to classify the text  [required]
  Input: [mutually_exclusive, required]
    -r, --repo_url URL            Github/Gitlab Repository URL
    -d, --doc_src PATH            Path to the README file source
    -i, --in_file PATH            A file of newline separated links to GitHub/
                                  Gitlab repositories

  Output: [required_any]
    -o, --output PATH             Path to the output file. If supplied, the
                                  output will be in JSON

    -c, --codemeta_out PATH       Path to an output codemeta file
    -g, --graph_out PATH          Path to the output Knowledge Graph export
                                  file. If supplied, the output will be a
                                  Knowledge Graph, in the format given in the
                                  --format option chosen (turtle, json-ld)
```

Image generated with Dall-e (https://labs.openai.com/)

Thoughts on making your software tool ready for publication

1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

- Use **logging**!!
- Errors are already frustrating. Silent errors will make your code **unusable**
- Meaningful error reporting will help you debug future errors

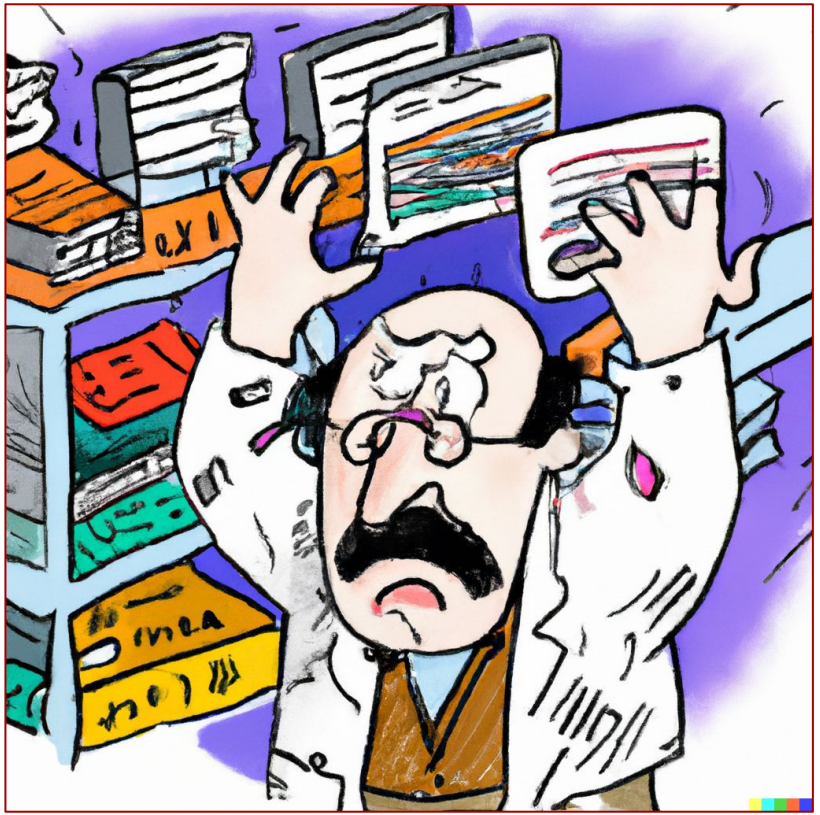


Image generated with Dall-e (https://labs.openai.com/)

1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

Without examples, your code becomes more difficult to run

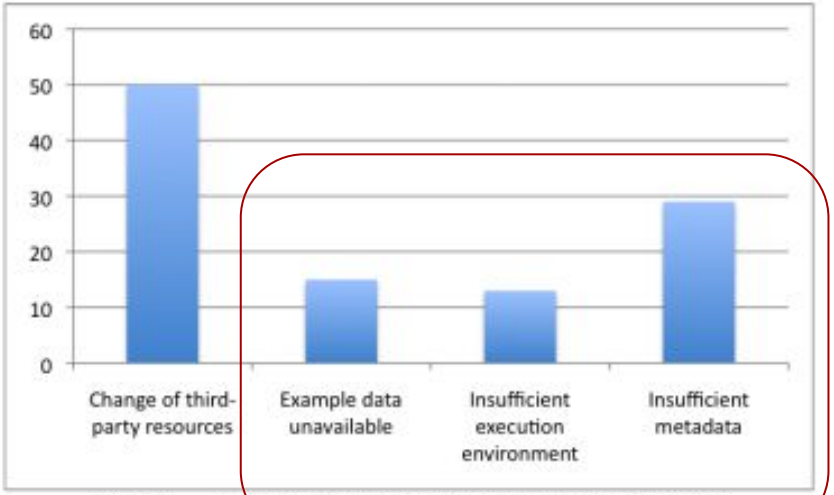


Fig. 9. A summary of workflow decay causes.

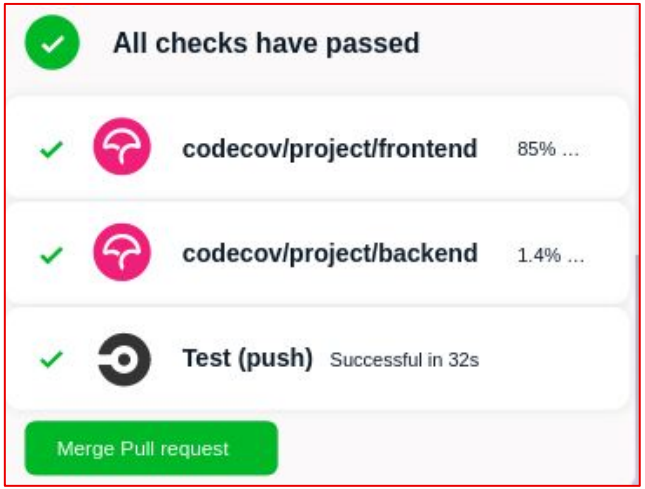
Why workflows break — Understanding and combating decay in Taverna workflows (Zhao et al 2012) 10.1109/eScience.2012.6404482.

1. Source code vs executable
2. Making software run elsewhere
3. Making software modular
4. Making software configurable
5. Making software report errors
6. Providing test data and examples
7. Code analysis

Many existing tools help you analyze your code/test coverage to improve the quality of your code

GitHub Copilot

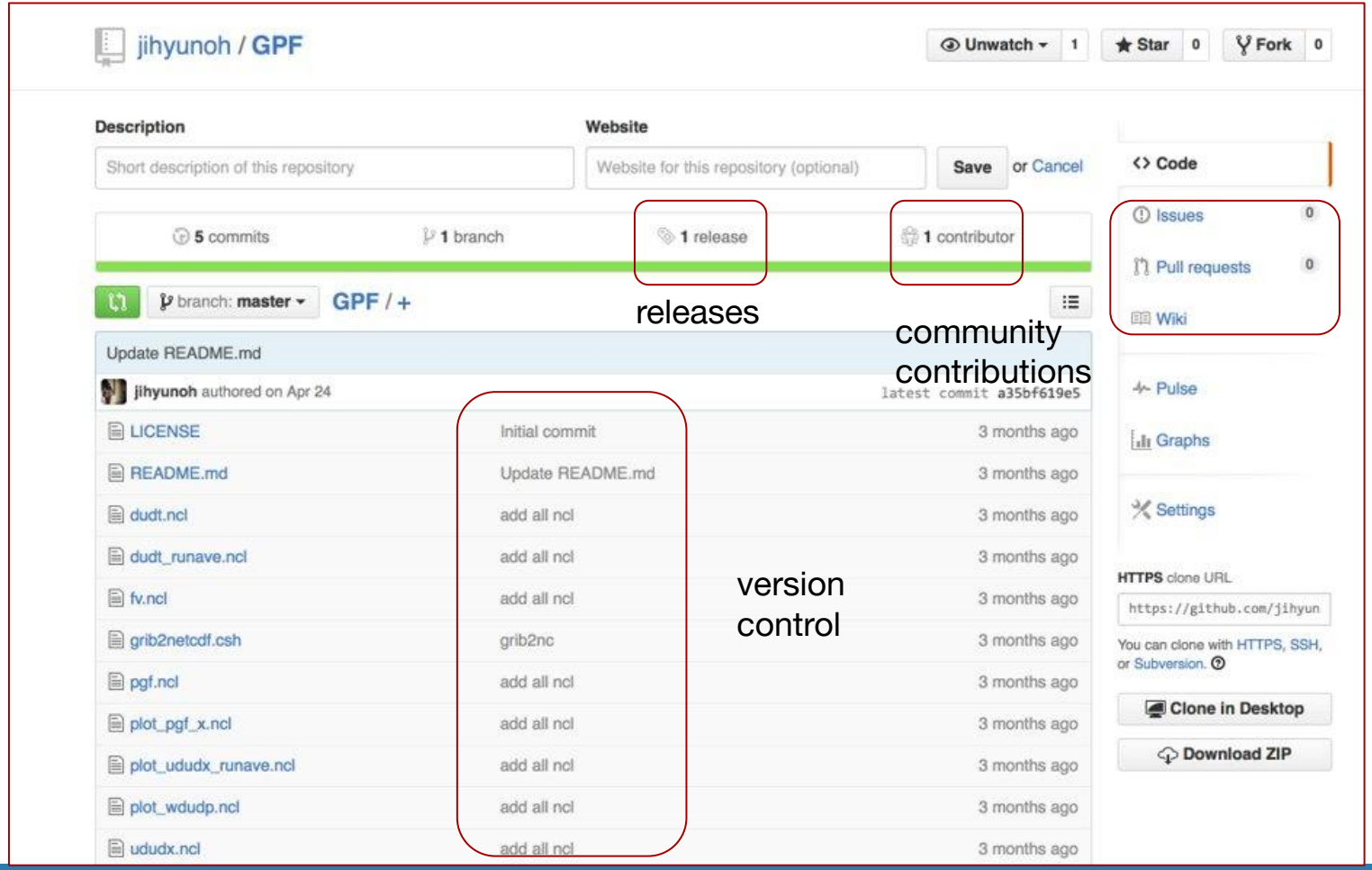
PyCharm



Best practices for making software accessible and reusable

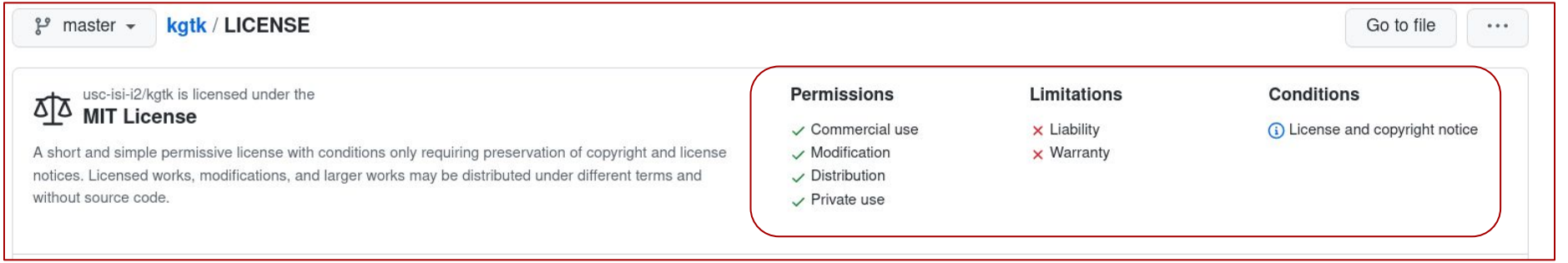
1. Use a public code repository
2. Declare a License
3. State a citation for your code
4. Describe your software with rich metadata
5. Create a comprehensive README file

Use a public code repository

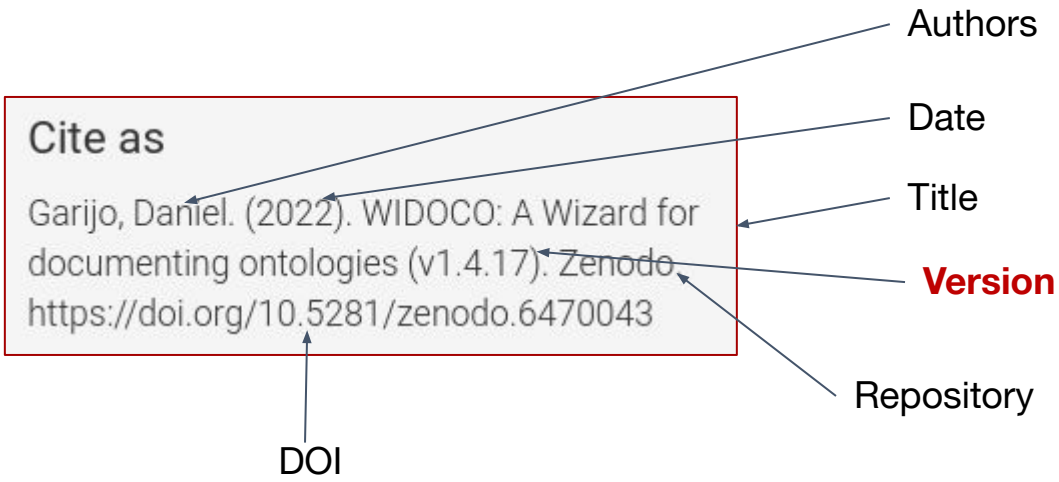


- **Copyright**: automatically applied to software when it is created to grant *the creator* exclusive rights as an intellectual property
- **Open source license**: reduce constraints and enable software developers to make their source code available to public
  - “Copyleft” license (ex: GNU General Public License (GPL))
  - “Permissive” license (ex: Apache 2 or MIT licenses)
- **Open Source Initiative**
  - Choose a license from: http://opensource.org/licenses
  - Recommend that you choose a permissive license
    - Apache v2

Some repositories help you choosing a license



- What do you want to cite?
  - Code? Project Website? Commit? Release?
- Use a **persistent unique identifier** (PURL or DOI)
  - Analogous to identifiers for data
- Software repositories are beginning to offer the ability to assign DOIs



Practical exercise: Obtain a DOI for your software releases

- Use your GitHub credentials to log into **Zenodo** (https://zenodo.org)
- **Authorize** Zenodo to access your GitHub account
- In **settings** -> GitHub, your repository should appear accessible
- Flip the switch to "**ON**"
- More details at https://guides.github.com/activities/citable-code/

Practical exercise: Obtain a DOI for your software releases

- Add code to your GitHub repository. When you are ready, click on “releases” and select “Create new release”.
- Describe your release
- Give it a proper **version number**!
- Now go to your Zenodo page. If everything went correctly, you should see a DOI for your GitHub repository
- Now you can copy the blue Zenodo badge with the DOI back in your GitHub readme file

- How should others cite your repository? Should they refer to a paper? Version?
- Citation File Format https://citation-file-format.github.io/
  - Helps stating how to cite a repository
  - Recently adopted by GitHub
  - Result of the FORCE 11 software citation WG
  - Makes your repository look nice!

How does a Citation file format work?)

```yaml
title: "SOMEF: Software metadata extraction framework"
license: Apache-2.0
authors:
  - family-names: Garijo
    given-names: Daniel
    orcid: "http://orcid.org/0000-0003-0454-7145"
  - family-names: Mao
    given-names: Allen
  - family-names: Dharmala
    given-names: Haripriya
  - family-names: Diwanji
    given-names: Cedant
  - family-names: Wang
    given-names: Jiajing
  - family-names: Kelley
    given-names: Aidan
  - family-names: García
    given-names: Miguel Angel
cff-version: 1.2.0
message: "If you use this software, please cite both the article from preferred-citation and the software itself."
preferred-citation:
  authors:
    - family-names: Kelley
      given-names: Aidan
    - family-names: Garijo
      given-names: Daniel
  title: "A Framework for Creating Knowledge Graphs of Scientific Software Metadata"
  type: article
  journal: "Quantitative Science Studies"
  pages: "1-37"
  year: 2021
  doi: 10.1162/qss_a_00167
```

- Title
- Authors
- Preferred citation (including version/doi)

Create your a CFF file for your course repository

Describing Research Software with rich metadata

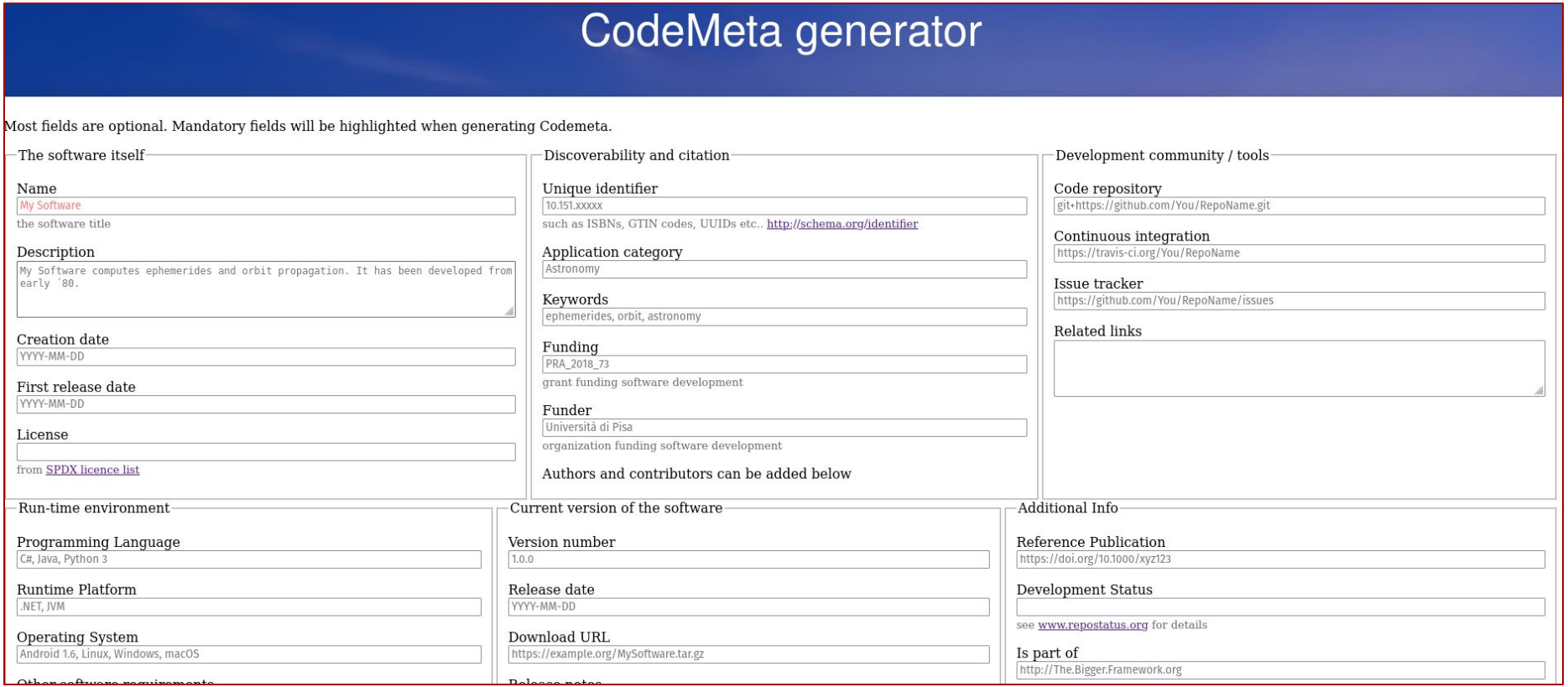
Any kind of software metadata can be useful to **find software**

- “I want **R code**…”
- “I want to see **software by** John Smith…”
- “I want software that is **well supported**…”
- “I want software that **simulates water runoff**…”
- “I want software that **uses elevation data**…”

Codemeta: community standard for software metadata

Property	Type	Description
softwareSuggestions	SoftwareSourceCode	Optional dependencies , e.g. for optional features, code development, etc.
maintainer	Person	Individual responsible for maintaining the software (usually includes an email contact address)
contIntegration	URL	link to continuous integration service
buildInstructions	URL	link to installation instructions/documentation
developmentStatus	Text	Description of development status, e.g. Active, inactive, suspended. See repostatus.org
embargoDate	Date	Software may be embargoed from public access until a specified date (e.g. pending publication, 1 year from publication)
funding	Text	Funding source (e.g. specific grant)
issueTracker	URL	link to software bug reporting or issue tracking system
referencePublication	ScholarlyArticle	An academic publication related to the software.
readme	URL	link to software Readme file

Schema.org extension (findable by search engines)



https://codemeta.github.io/codemeta-generator/ (manually) or https://somef.readthedocs.io/en/latest/ (automatically)

Create your own Codemeta JSON-LD file for your repository

A **README file** should include:

- Description
- Requirements
- Installation instructions
- Execution instructions
- Running example(s)
- Preferred citation (who is the main author?)
- Where to get help
- Acknowledgements (if any)

- **… there are many versions of the software?**
  - Give unique identifiers to the most significant versions that you want to release
  - Relate those versions to one another
- **… the software is already in a public repository?**
  - Use their preferred citation, and acknowledge their work
- **… the software is relatively small?**
- If you think it may be useful to someone (think of people who do not program!), then release it
- **… the software is a large package with many functions?**
  - Consider releasing the large package as a whole for those who want all the functionality
  - Consider also releasing pieces of it with limited functionality that may have

Summary: Making software accessible and reusable

- **Learn to use a public code repository**
  - GitHub, Gitlab, etc.
- **Create an entry, and track DOIs for releases**
  - Including a license -- choose from http://www.creativecommons.org/licenses
  - Create a comprehensive README file
  - Specify metadata (codemeta JSON-LD file)
    - Do not forget the version!
- **Specify desired citation (CFF file)**

And that 's it!

Analogous to citing data:

- Citation goes in the References section
- How to cite the software? You choose:
  - With an in-text pointer as you would cite any other paper (recommended)
  - With an in-text pointer in a special “Data Resources” (or “Software Resources”) section
  - With an in-text pointer in the “Acknowledgments” section

Best practices for

- Making Research Data accessible
- Making Research Software accessible

Using

- Shared repositories
- Persistent identifiers
- Rich metadata descriptions
- Open licenses

Preparing Deliverable 1

Start preparing for the first task (text extraction and analysis)

- Create an online GitHub repository for project deliverables
- Add an Open license
- Add a readme file
- Familiarize yourself with Grobid (https://github.com/kermitt2/grobid)
  - Text extraction tool from PDF
- Select up to 10 papers as your input dataset

Deadline **8th Feb, 2023**

Follow the best practices taught in class to perform an analysis over 10 open-access articles using Grobid (or other text analysis tools). Your program should:

- Draw a keyword cloud based on the abstract information
- Create a visualization showing the number of figures per article.
- Create a list of the links found in each paper.

You should explain (in your repository documentation) how you have validated each of your answers**. Create a document called "rationale.md" for this purpose**

Steps:

1) Make pipeline with Grobid and initial selection of papers
2) Create Python scripts for addressing the questions
    - **Deadline: Feb, 15th**
3) Create documentation and computational environment for running experiment
    - **Deadline: Feb, 22nd**
4) Dockerize experiment.

Deadline for full individual practice: March 8th**, 2023 (11:59 pm)**

Open Science and Artificial Intelligence in Research Software Engineering

Lecturer: Daniel Garijo

ETSI Informáticos, Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
https://oeg.fi.upm.es/

Session 2: Initiatives to describe Research Data and Software*

daniel.garijo@upm.es
@dgarijov

*With slides from "The Scientific Paper of the Future" training materials for the OntoSoft project (Gil et al)