Research Data Management

Dr. Sara El-Gebali

0000-0003-1378-5495
@yalahowy

SciLifeLab

Agenda

Overview of Research Data Management (RDM)

What is Data?

FAIR Principles

Where do we start?

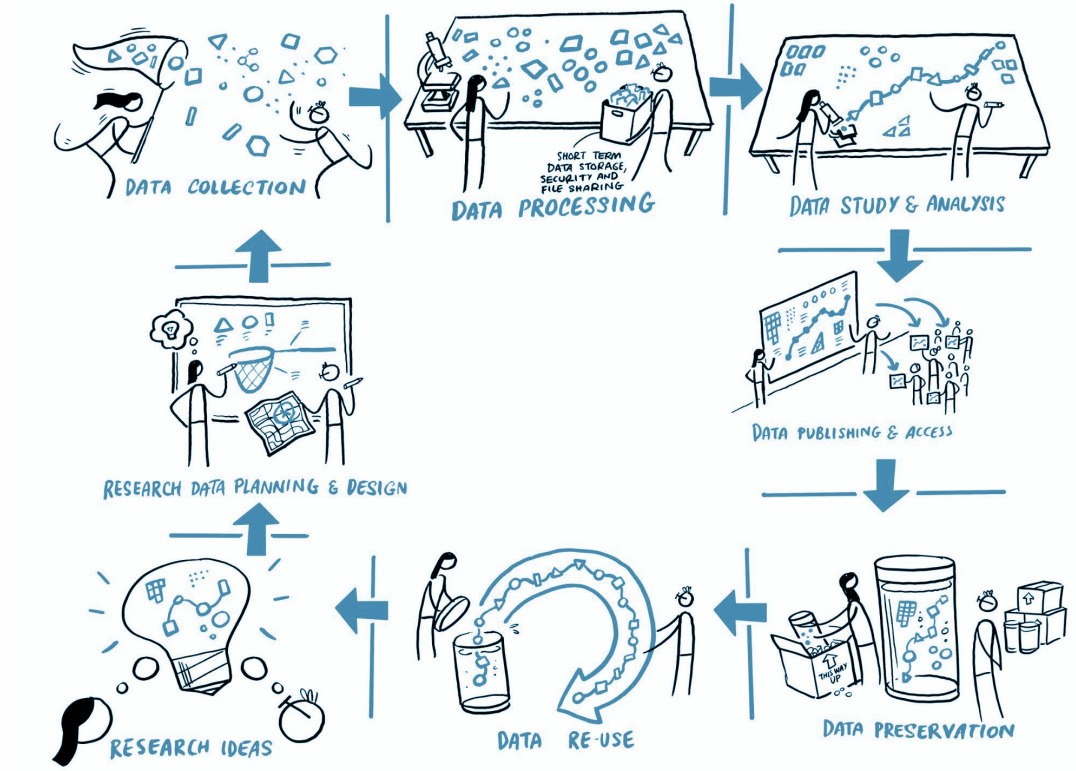
RDM best practices

FAIR vs Open

Open Data challenges

Research Data management

Everyday management of research data during the lifetime of a research project to preserve and share it beyond the project completion.



The Turing Way Community, & Scriberia. (2020, March 3). Illustrations from the Turing Way book dashes. Zenodo. http://doi.org/10.5281/zenodo.3695300

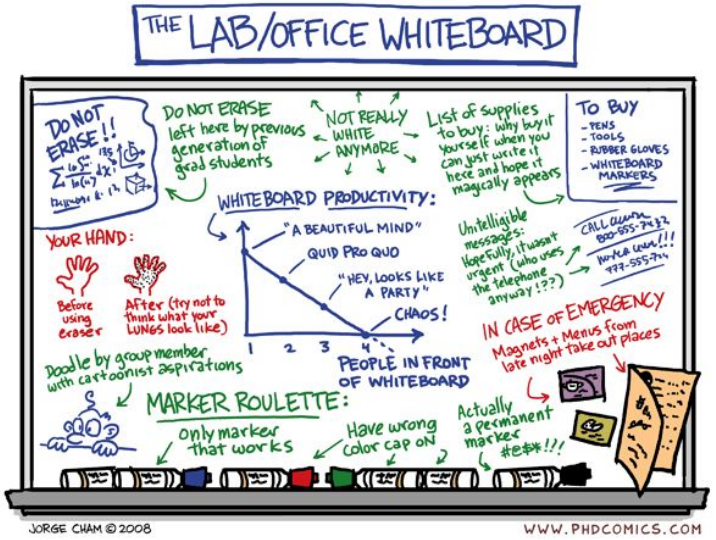
What is Data?

Primary- Raw from measurements or instruments

Secondary- Processed from secondary analysis and interpretations.

Published- final format available for use and reuse

Metadata- data about your data



It is everything that you need to validate or reproduce your research findings as well as what is required for the understanding and handling of the data.

Is Software Data?

In the chat:

- Type 1 for Data
- Type 0 for **NOT** Data

Research Software

" Software is data, but it is not just data. While "data" in computing and information science can refer to anything that can be processed by a computer, software is a special kind of data that can be a creative,executable tool that operates on data." 10.7287/peerj.preprints.2630v1

2. Software is not data

Technically, software is a special kind of data. In computing, digital data (ultimately sequences of ones and zeros) are used to represent all information, including factual data as well as computer instructions. In the more abstract context of FAIR, software and data are regarded as different kinds of digital research objects next to each other. As such, they share particular characteristics that allow them to be treated alike for certain aspects of FAIR, such as the possibility of having a Digital Object Identifier (DOI) assigned, or having a license. However, as elaborated by Katz et al. [4], there are also several significant differences between data and software as digital research objects: Data are facts or observations that provide evidence. In contrast, software is the result of a creative process that provides a tool for doing something, for example with data. As such, software is executable, while data is not. Software is often built using other software. This is especially

https://content.iospress.com/articles/data-science/ds190026

FAIR

The Turing Way Community, & Scriberia. (2020, March 3). Illustrations from the Turing Way book dashes. Zenodo. http://doi.org/10.5281/zenodo.3695300

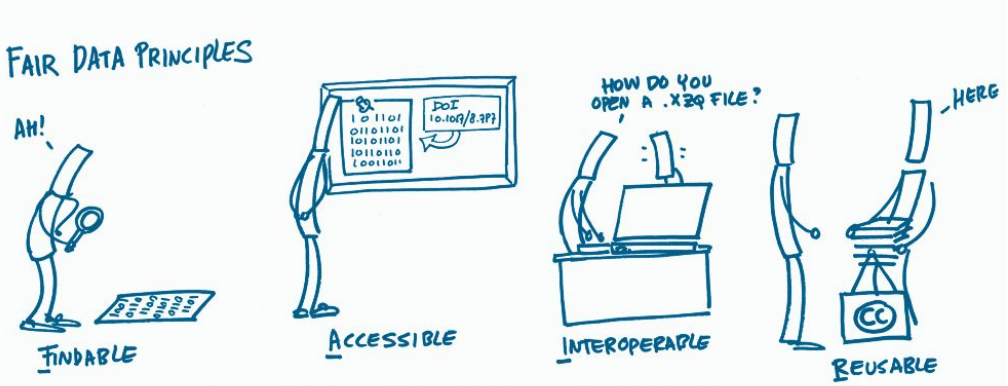
FAIR principles

FAIR is a set of principles to define the best practices for data and software to facilitate discovery, access and reuse by humans and machines.

FAIR is not rules and not a standard, it is an evolving process and a vision.

What does FAIR stand for?

Findable, Accessible, Interoperable and Reusable.



https://doi.org/10.5281/zenodo.1212496

Findable

Your data should be findable, have appropriate description (i.e. metadata), have a persistent identifier.

How to:

- Data and metadata should have a persistent identifier (a stable address where to find it), URL is not a PID.
- deposit your data to a domain-specific repository related to your field, https://www.re3data.org/
- Alternatively, deposit your data in general-purpose repositories such as Zenodo, Dryad, Dataverse.
- Same goes for your metadata.

https://doi.org/10.5281/zenodo.1212496

Accessible

Your data should be accessible for both humans and machines, i.e. retrievable and understandable

How to:

- Deposit the data under well defined conditions, i.e. data is accessible at HTTP or public REST API.
- Add clear licenses
- Specify what the users need to do to access this data,, i.e. two factor authentication, request access from author, etc…
- For private and sensitive data, the metadata (information about the data) can be made available and accessible.

https://doi.org/10.5281/zenodo.1212496

Remember

No license = No access!

‘As open as possible, as closed as necessary’

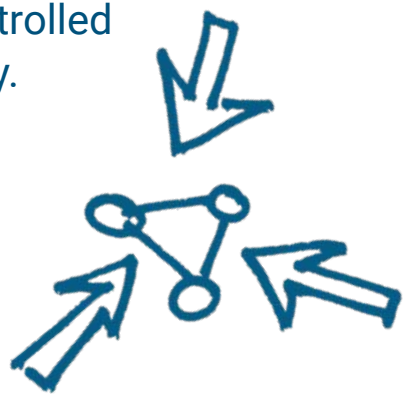
Even heavily protected and private data can be FAIR.

Interoperable

Machines and humans can interpret and use the data in different settings.

How to:

- Describe your data in detail - be generous!
- Describe your data properly, use controlled vocabulary, ontologies (controlled vocabulary with hierarchical relationships) and standardise terminology.
- Use preferred file formats, and open whenever possible.



https://doi.org/10.5281/zenodo.1212496

Reusable

The ultimate goal of FAIR is to advance the reuse of data. Everything you've done so far ultimately leads to this point, ensuring the data can be reused by others.

How to:

→ We will delve deeper into that!

https://doi.org/10.5281/zenodo.1212496

FAIR principles summary

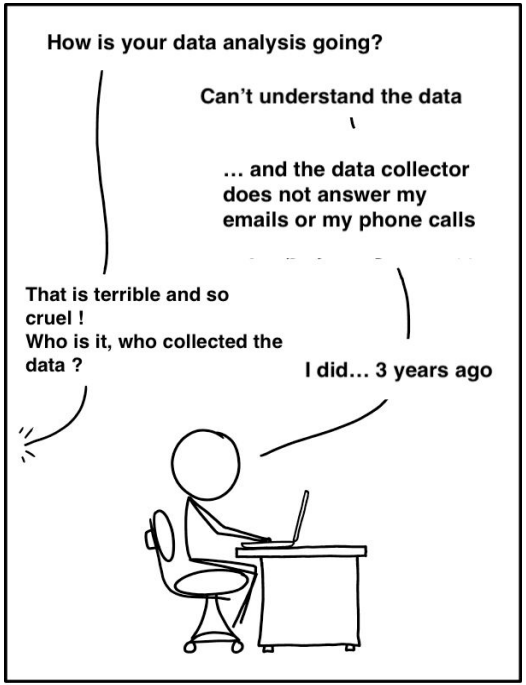
1. Deposit your data where others can find it, keep in mind where your peers can find it, i.e. field specific repository and give it a stable unique identifier (PID).
2. Make your data & metadata accessible via standard means such as http/API.
3. Create metadata and explain in detail what this data is about, never assume people know!
4. Deposit metadata with PID and make it available with/out data i.e. in case data itself is heavily protected.
5. Include information on ownership and provenance.
6. Outline what the reusers of your data are/not allowed to do, use clear license. Commonly used licenses like MIT or Creative Commons (keep in mind funders requirements).
7. Specify access conditions, if authentication or authorization is required.
8. Describe your data in a standardized fashion using agreed terminology and vocabulary.
9. Share the data in preferred & open file formats.

10. Start the process early on!

Why should make my data reusable?

You are the first one to reuse your data. Do you understand what you did a year ago?

You are not alone! Research relies on collaboration, can your collaborators understand what you did?



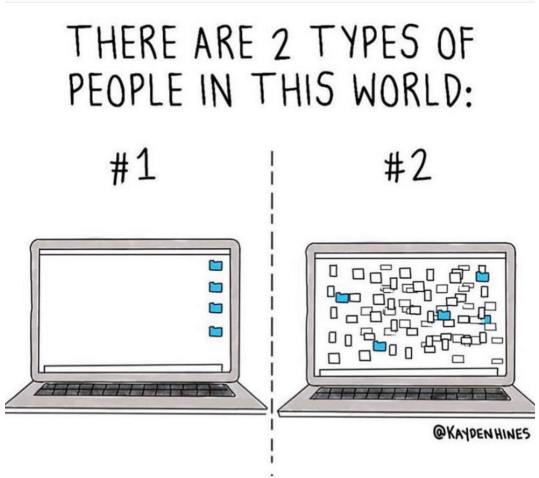
Your first collaborators are your future selves, be nice to them !

Organize your data

Organize your data in a logical manner

Separate the data according to type: i.e. raw data, analysis, code,

Use directories and folders hierarchy

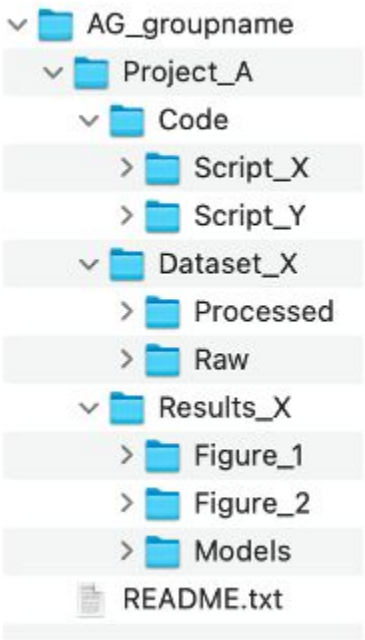


A clear directory structure will make it easier to locate files and versions and this is particularly important when collaborating with others.

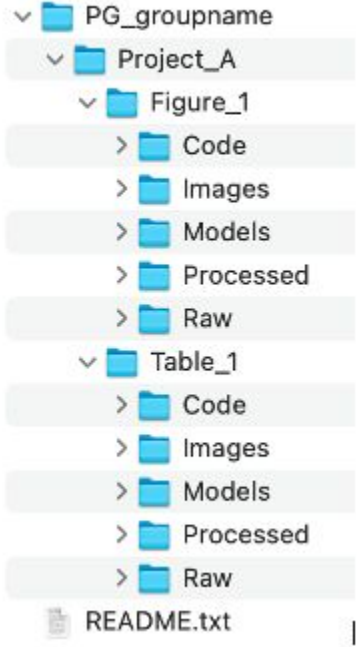
Directory structure guidelines

Consider a hierarchical file structure starting from broad topics to more specific ones nested inside, restricting the level of folders to 3 or 4 with a limited number of items (max. 50 items if possible) inside each folder.

A) Organized by file type



B) Organized by analysis



Name your files

Common guideline, you should know what your file is before you double click it!

A file name should be unique, consistent and descriptive.

This allows for increased visibility and discoverability and can be used to easily classify and sort files.

Choose file formats wisely

- During your research, your choice of file formats might be dictated by convenience or instrument provider or team practices.
- To make your data available for others and easy to use, choose file formats that are most commonly used in your field (e.g. fasta/fastq) or open file formats that allow interoperability (e.g. mark down)

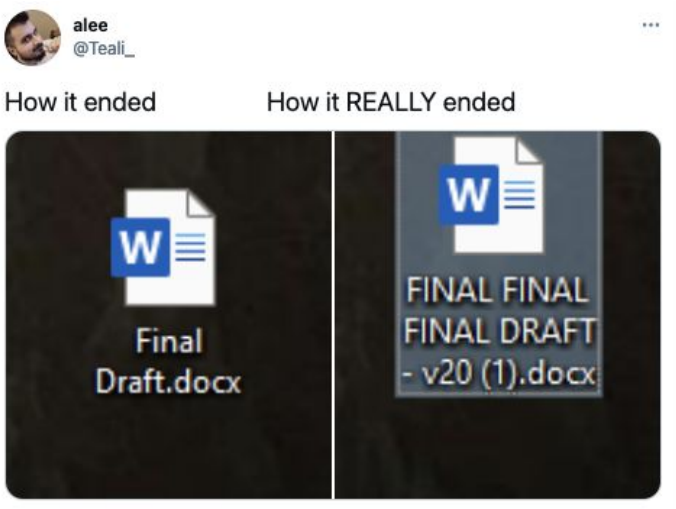
	Preferred formats	Accepted formats
Text documents	• ASCII (.txt) • MS Word (.docx) • OpenDocument Text (.odt) • PDF/A (.pdf) • Unicode (.txt)	• MS Word (.doc) • PDF (.pdf) • Rich Text Format (.rtf)
Markup language	• HTML (.html) • JSON (.json) • XML (.xml)	• SGML (.sgml) • Markdown (.md)
Spreadsheets	• CSV (.csv) • MS Excel (.xlsx) • OpenDocument Spreadsheet (.ods)	• MS Excel (.xls) • OOXML (.docx, .docm) • PDF/A (.pdf)
Databases	• CSV (.csv) • SIARD (.siard) • SQL (.sql)	• dBase III or IV (.dbf) • Filemaker Pro (.fmp7, .fmp12) • MS Access (.mdb, .accdb) • OpenDocument Base (.odb)
Statistical data	• OpenDocument (.ods) • SPSS portable (.por) • SPSS SAV (.sav) • STATA (.dta)	• CSV (.csv) • MS Excel (.xls, .xlsx) • R (.rdata, .rda) • SAS (.sas) • SAS transport (.xpt)
Image (raster/bitmap)	• Adobe Digital Negative format (.dng) • DICOM (.dcm) • PNG (.png) • TIFF (.tif, .tiff)	• Adobe Photoshop document file (.psd) • JPEG (.jpg, .jpeg) • JPEG 2000 (.jp2, .jpx) • Raw image data (various formats)

https://snd.gu.se/en/manage-data/guides/suggested-file-formats

Versioning

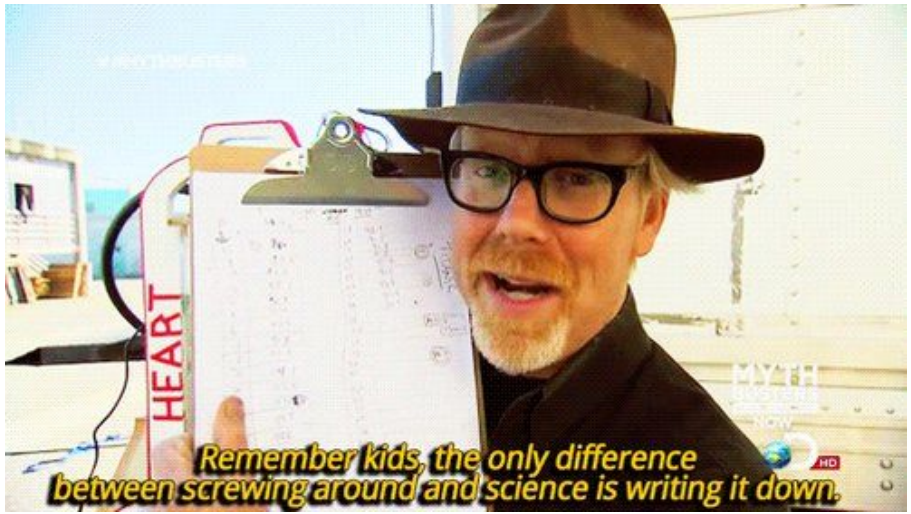
In order to keep track of changes made to a file/dataset, versioning can be an efficient way to see who did what and when, in collaborative work this can be very useful.

A version control strategy will allow you to easily detect the most current/final version, organize, manage and record any edits made while working on the document/data, drafting, editing and analysis.



What should I include? METADATA

- **Who** created and owns this data?
- **What** are the contents?
- **What** output and results?
- **When** was this data created and last updated?
- **Where** is it stored and published?
- **Which** methods were used?
- **Which** instruments were used?
- **How** was the data created, controlled and analysed?
- **How** can I use this data i.e. license?



https://www.tested.com/making/557288-origin-only-difference-between-screwing-around-and-science-writing-it-down/

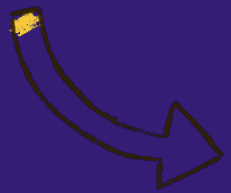
FAIR ≠ Open

‘FAIR is not the equivalent of open, but open needs to be FAIR to be useful’

Making your data/software freely and openly available does not translate to it being reusable!

To do so, we need clear, detailed contextual information and data description.

Data can be FAIR but not Open! FAIR data motto “as open as possible, as closed as necessary”



Ideally you want FAIR data/software shared openly!

What challenges do you face sharing your data openly?

Challenges for open data

I will be scooped!

With very little evidence to support this notion, sharing your data openly, with appropriate licensing ensures your claim of authorship.

This has been one of the major driving factors for the publications of preprints.

Benefits of Preprints

We see preprints as an important step toward a more open and transparent peer review process — one with tremendous benefits for both individual authors and the broader scientific community.

Rapid Dissemination of Your Results

Preprints allow you to share your results when you're ready — whether you're researching an emergent disaster, applying for a grant, or just excited to broadcast your work to a wider audience.

Establishing Priority

It's common for researchers to achieve a similar advance at around the same time but the publication process can artificially delay one paper or favor another. Posting preprints allows researchers to publicly date stamp their discoveries.

Increased Attention (and Citations!)

The sooner research becomes available, the sooner it can begin to receive views and citations. In this case, common sense is backed up by evidence. Research shows that public posting increases the number of times papers are viewed and cited.

https://plos.org/open-science/preprints/

Challenges for open data

What if people misinterpret my data?

Clear and detailed documentation is key! Include rich metadata and outline the restrictions to using this data.

Include your ORCID details to allow a chance for the data reuser to get in touch and inquire before making any judgements or misinterpretations.

Papers Without Code - submission

Papers Without Code: where unreproducible papers come to live.

The goal of this is to save the time and effort of researchers who try to reproduce the results of a paper that is unreproducible. It could either be due to the paper not having enough details or the method straight up not working. In either case, authors will be given the opportunity to respond. The hope is this saves people time and disincentivizes unreproducible papers.

1. First authors of the paper will be informed and given a chance to respond.

2. Submissions with multiple votes and/or a link to a reproduction will be given priority.

3. Every submission will be reviewed to prevent spam. If it is a genuine submission, expect it to be approved within 24 hours.

Note: In order to protect the authors reputation, we will take spamming very seriously.

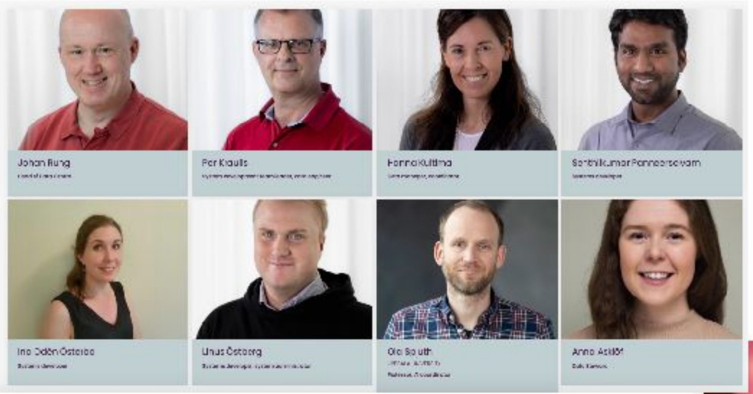
Contact: papersburned@gmail.com
Results: papers.paperswithoutcode.com

https://www.paperswithoutcode.com/

Find out more

https://zenodo.org/record/4562630

SciLifeLab


Johan Rung
Per Kraulis
Hanna Kultima
Senthilkumar Panneerselvam
Ina Odén Österbo
Linus Östberg
Ola Spjuth
Anna Asklöf


Arnold Kochari
Katarina Öjefors Stark
Parul Tewatia
Liane Hughes
Hamza Imran
Sara El-Gebali
Matteo Carone
Volkan Cambazoglu

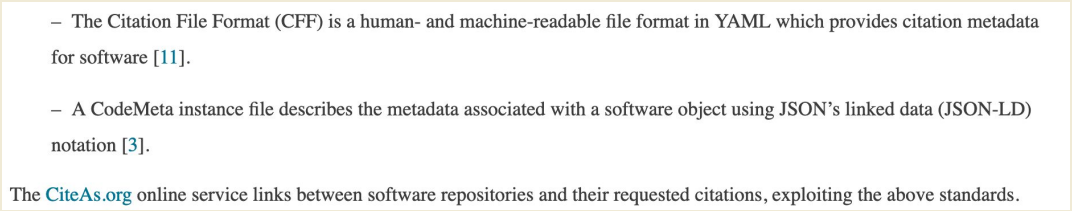
The End

FAIR Software

Findable- Software

- Indicate all versions of the software in the metadata and how to find them.

- Use of metadata standards for software citation and identification



– The Citation File Format (CFF) is a human- and machine-readable file format in YAML which provides citation metadata for software [11].

– A CodeMeta instance file describes the metadata associated with a software object using JSON's linked data (JSON-LD) notation [3].

The CiteAs.org online service links between software repositories and their requested citations, exploiting the above standards.

https://doi.org/10.1515/itit-2019-0040

- Choice of repositories includes;
  - **General** repositories such as Zenodo and Github,
  - **Language** specific, Python Package Index (PyPI) is a repository of software for the Python programming language.https://pypi.org/
  - **Domain** specific; https://biocontainers.pro/

Accessible- Software

Metadata should be accessible separately from the software and available even if the software isn't.

For example: metadata can be deposited on Zenodo with/out snapshot of the software with/out versioned software on Github

Different versions of the software should have different PIDs.

How does DOI versioning work?

When you publish an upload on Zenodo for the first time, we register two DOIs:

- a DOI representing the **specific version** of your record.
- a DOI representing **all of the versions** of your record.

Afterwards, we register a DOI for every new version of your upload.

This is best illustrated by an example of a software package. If the software has been released in two versions (v1.0 and v1.1) on Zenodo, then the following DOIs would have been registered:

- **v1.0 (specific version):** 10.5281/zenodo.60943
- **v1.1 (specific version):** 10.5281/zenodo.800648
- **Concept (all versions):** 10.5281/zenodo.705645

The first two DOIs for versions **v1.0** and **v.1.1** represent the specific versions of the software. The last DOI represents all the versions of the given software package, i.e. the concept of the software package and the ensemble of versions. We therefore also call the them **Version DOIs** and **Concept DOIs** (note, technically both are just normal DOIs).

You may notice that the version DOIs do not include a ".v1"-suffix. Read below to find out why.

https://help.zenodo.org/

Interoperable- Software

Interoperability is the biggest challenge!

- Containers (e.g. use Docker, singularity) allows for accessibility across different operating systems and environments i.e. software portability.
- Rich metadata is key!
- The use of Common Workflow Language (CWL) enables the interoperability between different pieces of software and workflow platforms

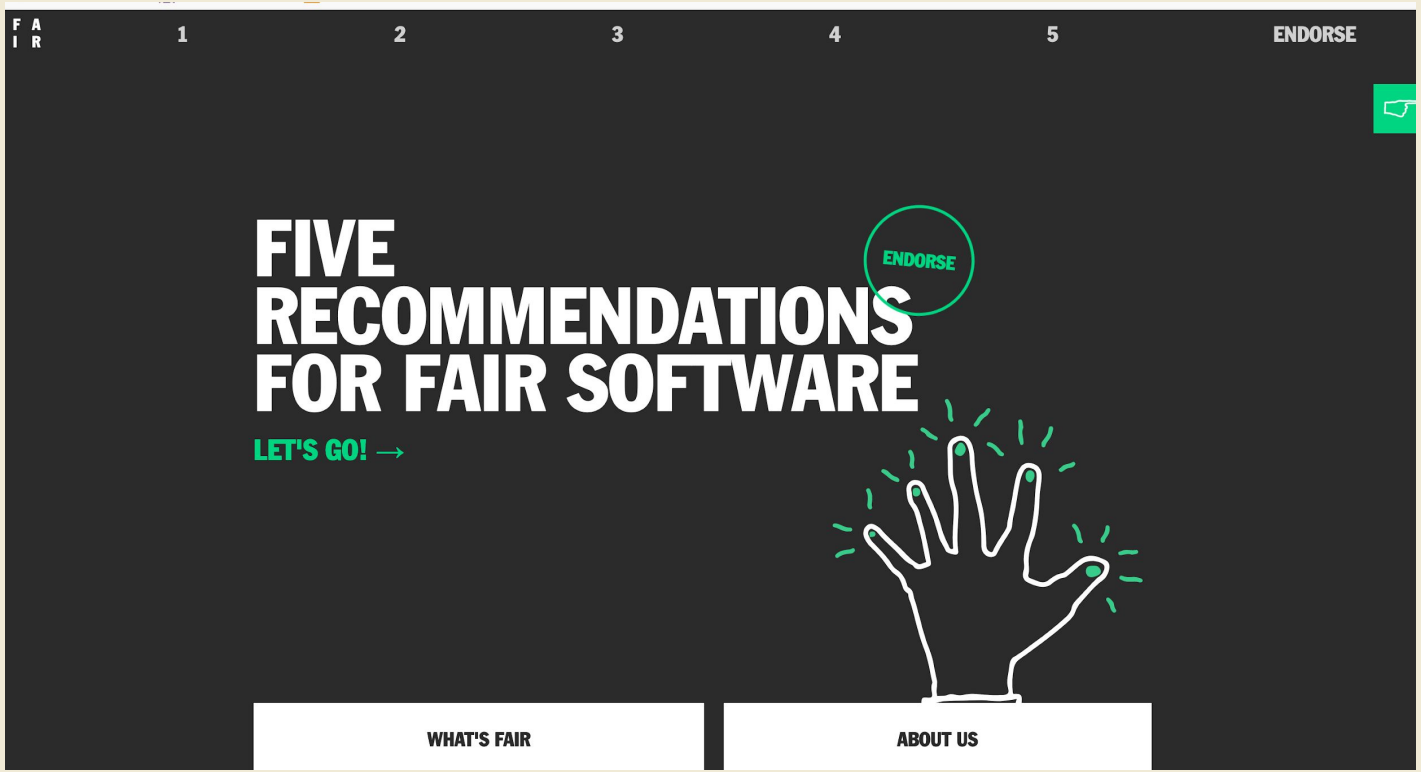
https://doi.org/10.1093/gigascience/giz095 // https://www.commonwl.org/user_guide/01-introduction/index.html

Interoperable- Software

- Controlled vocabulary:
  - The Software Ontology
  - Codemeta and enables proper citation
  - Disciple specific ontologies such as EDAM - Ontology of bioscientific data https://edamontology.org/page
  - More domain specific controlled vocabulary/ontologies can be found at FAIRsharing.org

FAIR Software

https://fair-software.eu/



FAIR software summary

1. Deposit in publicly accessible repositories https://software.ac.uk/choosing-repository-your-software-project
2. Use a version control system to easily track changes and versions, Github, Gitlab, Bitbucket,
3. Use of containers for software portability
4. Describe with rich metadata including dependencies, with controlled vocabulary
5. Explain the intended use and conditions of functionality of the software
6. Add a license, Apache-2.0 and MIT are permissive licenses with few restrictions, allowing reuse. https://choosealicense.com/ // https://tldrlegal.com/
7. Register your code in a community https://github.com/NLeSC/awesome-research-software-registries
8. Store snapshots of your software with PIDs https://guides.github.com/activities/citable-code/
9. Enable proper citation for your software, CodeMeta and the Citation File Format were specifically designed to enable citation of software
10. Quality control; use of checklists and include as part of the README file https://github.com/eurise-network/technical-reference/blob/v0.1/quality/software-checklist.rst

Software- Resources

Reproducible analysis and Research Transparency
Data Science for the Biomedical Sciences
From FAIR research data toward FAIR and open research software
Towards FAIR principles for research software
Software vs. data in the context of citation
Assessment report on 'FAIRness of software'
Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv
Common Workflow Language User Guide
FAIR Software
Python package to analyze a GitHub or GitLab repository's compliance with the fair-software.eu recommendations.
Checklist for a Software Management Plan
Top 10 FAIR Data & Software Things
Research Software Alliance
CodeMeta
Open source for open science- CERN
Software Reproducibility - The Nuts and Bolts
Is software reproducibility possible and practical?
Make a README
README awesome list

Tools

Tools for bulk renaming

Windows:

- Ant Renamer (www.antp.be/software/renamer)
- Bulk Rename Utility (www.bulkrenameutility.co.uk/)
- Total Commander (https://www.ghisler.com/deutsch.htm)

Mac:

- Renamer 5 (for Mac) (https://renamer.com/)
- Name Changer (https://mrrsoftware.com/namechanger/)
- ExifRenamer (https://www.qdev.de/?location=mac/exifrenamer)

Linux:

- GNOME Commander (www.nongnu.org/gcmd/)
- GPRename (http://gprename.sourceforge.net/)

Unix:

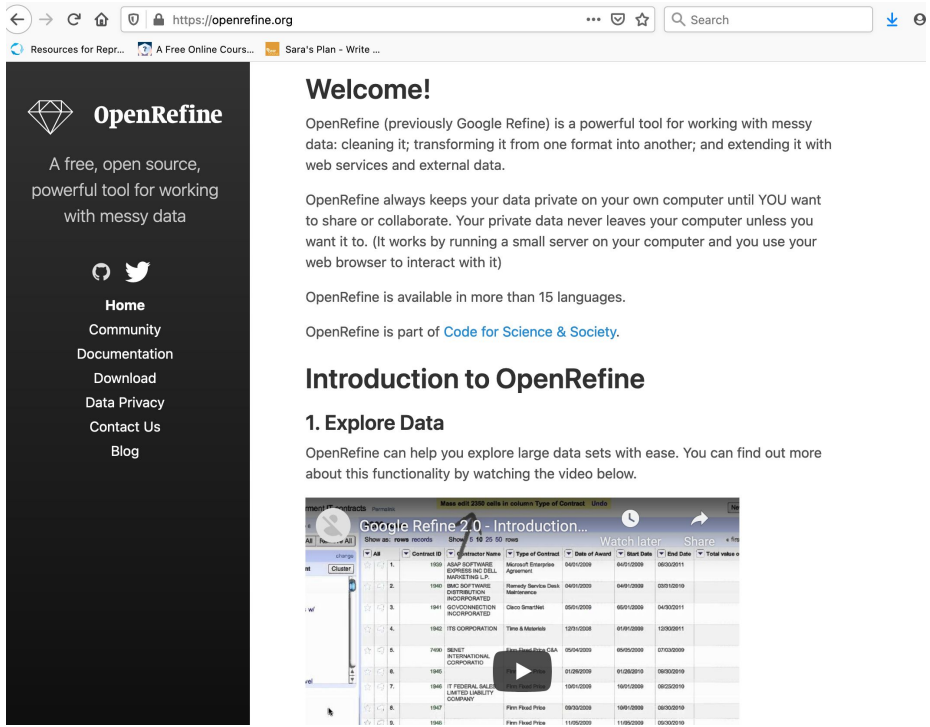
- Rename command

https://doi.org/10.5281/zenodo.1212496

Tools for Quality control- Qualitative data

Tools

- Open Refine for data quality control https://openrefine.org/
- Numeric data anonymization R-Package: sdcMicro
- UK data archive tools list: https://www.data-archive.ac.uk/managing-data/digital-curation-and-data-publishing/tools-we-use/



Metadata- README - software

https://www.makeareadme.com/

README 101

What is it?

A README is a text file that introduces and explains a project. It contains information that is commonly required to understand what the project is about.

Why should I make it?

It's an easy way to answer questions that your audience will likely have regarding how to install and use your project and also how to collaborate with you.

Who should make it?

Anyone who is working on a programming project, especially if you want others to use it or contribute.

When should I make it?

Definitely before you show a project to other people or make it public. You might want to get into the habit of making it the first file you create in a new project.

Where should I put it?

In the top level directory of the project. This is where someone who is new to your project will start out. Code hosting services such as GitHub, Bitbucket, and GitLab will also look for your README and display it along with the list of files and directories in your project.

How should I make it?

While READMEs can be written in any text file format, the most common one that is used nowadays is

Suggestions for a good README

Every project is different, so consider which of these sections apply to yours. The sections used in the template are suggestions for most open source projects. Also keep in mind that while a README can be too long and detailed, too long is better than too short. If you think your README is too long, consider utilizing another form of documentation rather than cutting out information.

Name

Choose a self-explaining name for your project.

Description

Let people know what your project can do specifically. Provide context and add a link to any reference visitors might be unfamiliar with. A list of **Features** or a **Background** subsection can also be added here. If there are alternatives to your project, this is a good place to list differentiating factors.

Badges

On some READMEs, you may see small images that convey metadata, such as whether or not all the tests are passing for the project. You can use Shields to add some to your README. Many services also have instructions for adding a badge.

Visuals

Depending on what you are making, it can be a good idea to include screenshots or even a video (you'll frequently see GIFs rather than actual videos). Tools like ttygif can help, but check out Asciinema for a more sophisticated method.

Installation

Tools for versioning & Packaging

Tools

- Singularity: https://github.com/hpcng/singularity
- Docker: https://www.docker.com/resources/what-container
- Jupyter : https://jupyter.org/index.html
- Fido: https://github.com/openpreserve/fido
- Vagrant: https://www.vagrantup.com/intro/vs/docker.html

101 innovations

https://101innovations.wordpress.com/