FAIR Software Management

How to handle research software

Yvana Glasenapp, LUH Research Service

Anna-Karina Renziehausen, TIB Publication Services

18.07.2024

Welcome!

Anna-Karina Renziehausen

TIB - Leibniz-Informations-zentrum Technik und Naturwissenschaften und Universitätsbibliothek

Publikationsdienste

Das Service-Team Forschungsdaten

www.fdm.uni-hannover.de/en

visit the website of the Research Data Support Team

Yvana Glasenapp

Gottfried Wilhelm Leibniz Universität Hannover

Dezernat Forschung und EU-Hochschulbüro, Technologietransfer

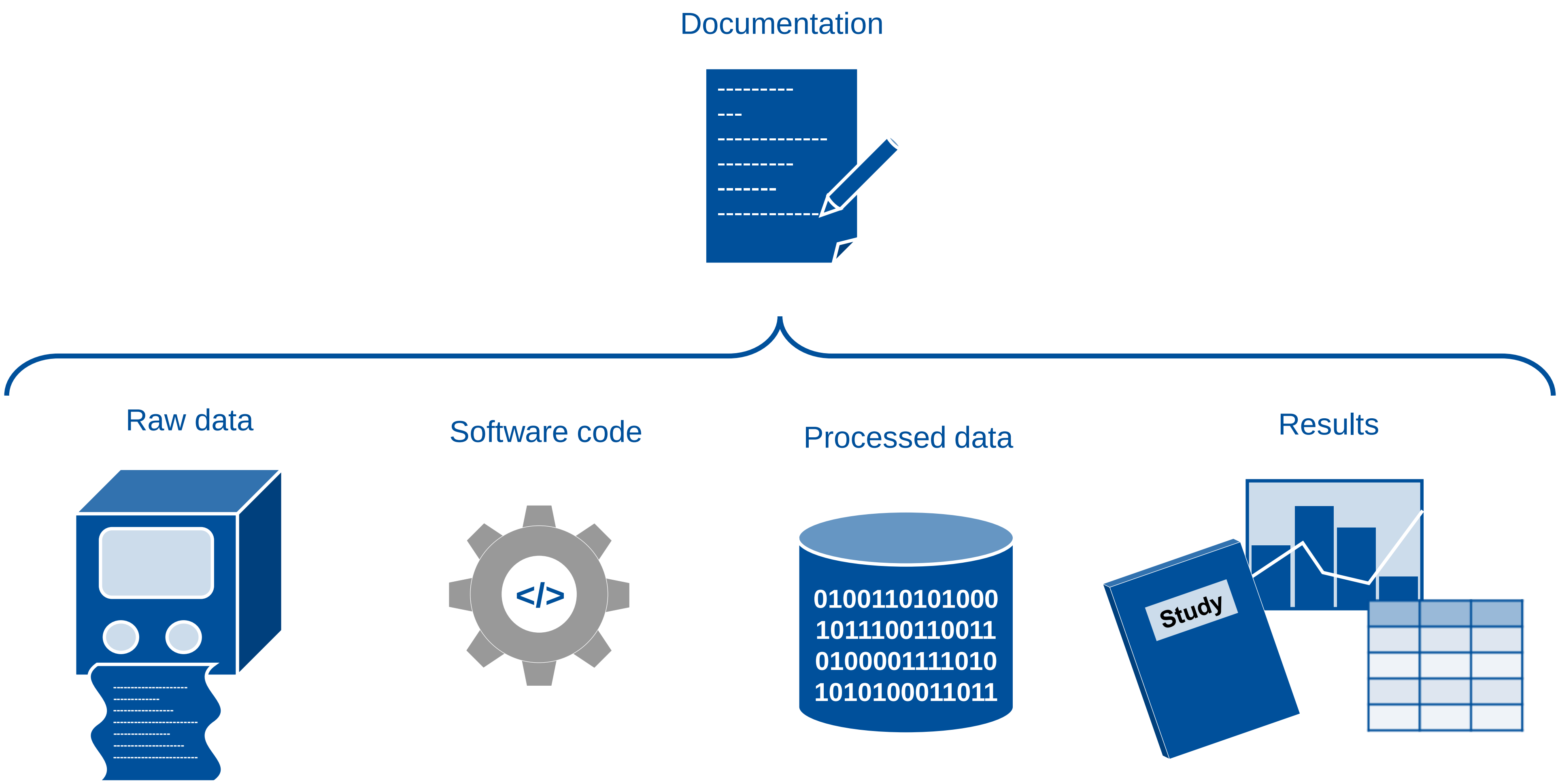
Content of this course

- Introduction to Research Data Management
- How to make research software re-usable
- How to publish research software and code
- Supporting services and initiatives

Introduction

- What are research data?
- What is research data management (RDM)?
- Why is data management important?
- The FAIR principles
- RDM is good scientific practice!
- Guidelines for handling research data at LUH
- LUIS services for data storage and backup

What are research data?



Documentation

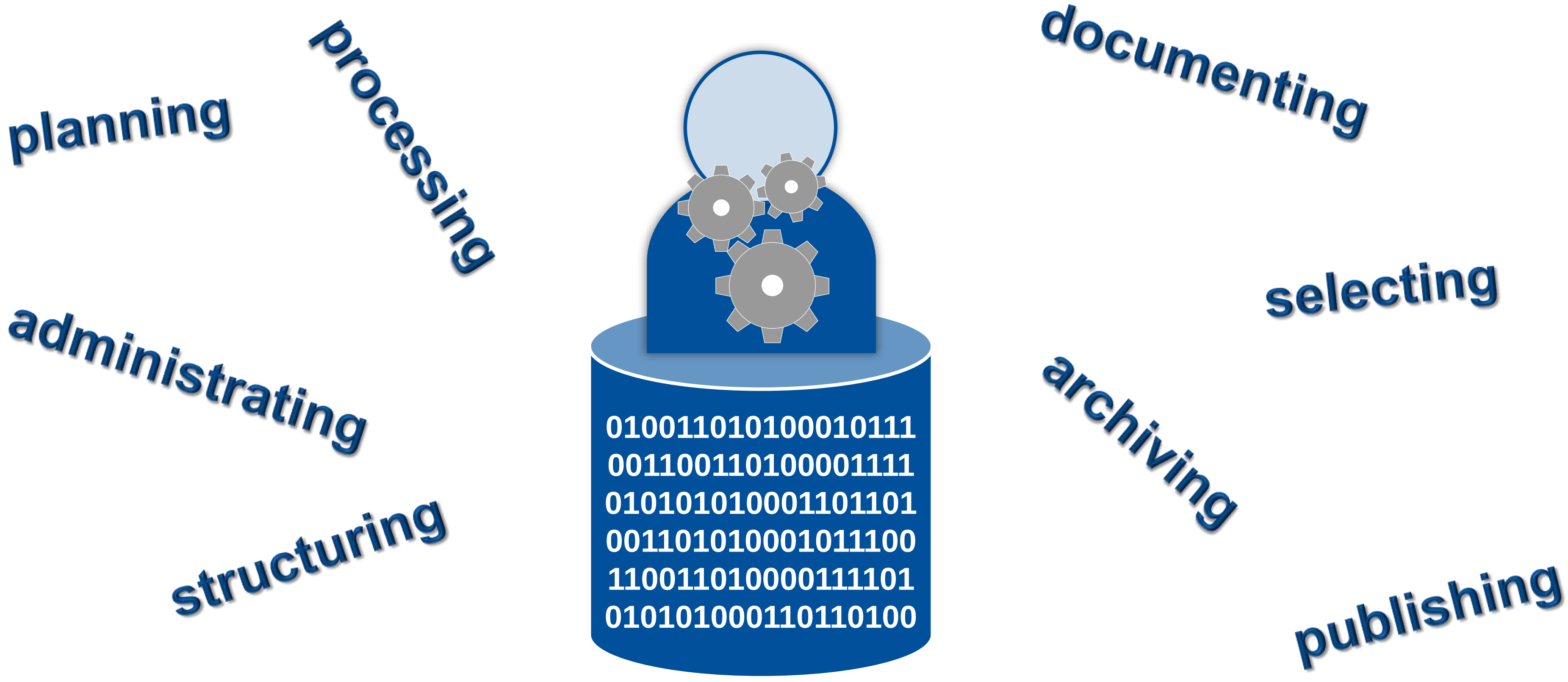
Raw data

Software code

Processed data

Results

What is research data management (RDM)?



planning

processing

documenting

selecting

administrating

archiving

structuring

publishing

i

RDM: Conscious and systematic handling of data from the planning stage until the end of a project and perhaps even beyond.

Why is data management important?

✓ you keep an overview

✓ team work is easier

✓ you can safeguard high quality standards in research

✓ you save time and avoid stress

✓ you comply with official requirements

✓ Data management is the basis for Open Science

✓ You gain recognition for published data and software

further reading

The Thuringian Competence Network for Research Data Management compiled some “Research Data Scarytales”. These are true stories about data management failures and their consequences.

go to the „Scarytales“

The FAIR Principles

Findable

- Unique and unambiguous identifier (e.g. DOI)
- Rich metadata (containing the DOI)
- Indexing in searchable directories (e.g. data repo)

Accessible

- Accessible via a standardized, open, free, universal communication protocol
- Transparent access (authentication)
- Metadata accessible even if data is not

Interoperable

- Standardized and widely applicable language for knowledge representation
- Vocabulary that follows FAIR principles
- References to related data

Re-usable

- Good documentation and precise attributes
- Clear license
- Detailed provenance
- Community standards

GO FAIR

GO FAIR Website

RDM is good scientific practice!

RDM is especially relevant for these 6 out of 19 guidelines:

Guideline 7: Cross-phase quality assurance

Guideline 10: Legal and ethical frameworks, usage rights

Guideline 11: Methods and standards

Guideline 12: Documentation

Guideline 13: Providing public access to research results

Guideline 17: Archiving

i

You can read the guidelines themselves and accompanying disciplinary comments in the DFG portal “Research Integrity” (English version not finished, yet).

to the DFG portal

!

Please note as well:

- DFG guidelines on the handling of research data
- disciplinary guidelines, requirements, policy papers etc.

You can find both on this website:

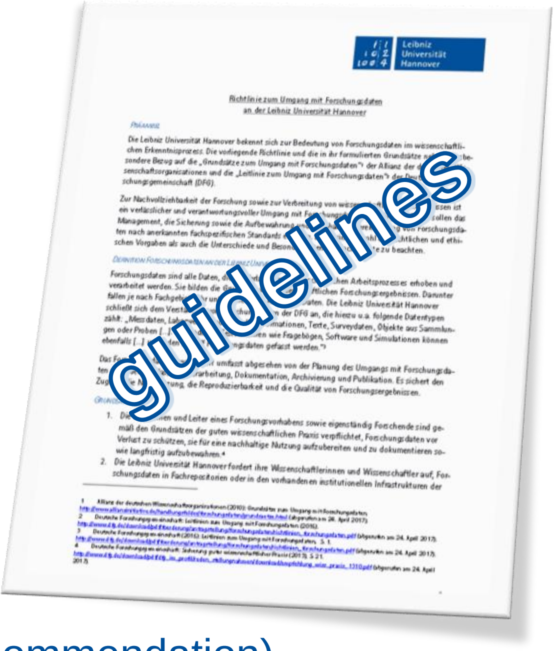
to the DFG website „Handling of Research Data“

Guidelines for handling research data at LUH

The four principles:

- Data management:
  - protect against losses
  - process for sustainable (re-)use
  - document
  - archive (long-term)
- publish research data according to the FAIR principles in (disciplinary) repositories
- develop project-internal RDM policies and data management plans (recommendation)
- integrate RDM into teaching (recommendation)

→ These principles do not include specific advice on research software (yet). But they do apply for software, as it is listed as a type of research data in the definition



Leibniz Universität Hannover

Richtlinie zum Umgang mit Forschungsdaten an der Leibniz Universität Hannover

PRÄAMBEL

Die Leibniz Universität Hannover bekennt sich zur Bedeutung von Forschungsdaten im wissenschaftlichen Erkenntnisprozess. Die vorliegende Richtlinie und die in ihr formulierten Grundsätze n[illegible] besondere Bezug auf die „Grundsätze zum Umgang mit Forschungsdaten“[1] der Allianz der d[illegible]senschaftsorganisationen und die „Leitlinie zum Umgang mit Forschungsdaten“[2] der Deut[illegible]schungsgemeinschaft (DFG).

Zur Nachvollziehbarkeit der Forschung sowie zur Verbreitung von wiss[illegible]ssen ist ein verlässlicher und verantwortungsvoller Umgang mit Fo[illegible] sollen das Management, die Sicherung sowie die Aufbewahrung u[illegible] von Forschungsdaten nach anerkannten fachspezifischen Standards [illegible]ohl [illegible]chtlichen und ethischen Vorgaben als auch die Unterschiede und Beson[illegible]te zu beachten.

DEFINITION FORSCHUNGSDATEN AN DER LEIBNIZ UNIV[illegible]

Forschungsdaten sind alle Daten, di[illegible]chen Arbeitsprozesses erhoben und verarbeitet werden. Sie bilden die G[illegible]ftlichen Forschungsergebnissen. Darunter fallen je nach Fachgebi[illegible]daten. Die Leibniz Universität Hannover schließt sich dem Vers[illegible]n der DFG an, die hiezu u.a. folgende Datentypen zählt: „Messdaten, Lab[illegible]mationen, Texte, Surveydaten, Objekte aus Sammlungen oder Proben [...] [illegible]en wie Fragebögen, Software und Simulationen können ebenfalls [...] [illegible]gsdaten gefasst werden.“[3]

Das Fo[illegible] umfasst abgesehen von der Planung des Umgangs mit Forschungsdaten [illegible]rarbeitung, Dokumentation, Archivierung und Publikation. Es sichert den Zug[illegible]ung, die Reproduzierbarkeit und die Qualität von Forschungsergebnissen.

GRUND[illegible]

1. Die[illegible]nen und Leiter eines Forschungsvorhabens sowie eigenständig Forschende sind gemäß den Grundsätzen der guten wissenschaftlichen Praxis verpflichtet, Forschungsdaten vor Verlust zu schützen, sie für eine nachhaltige Nutzung aufzubereiten und zu dokumentieren sowie langfristig aufzubewahren.[4]
2. Die Leibniz Universität Hannover fordert ihre Wissenschaftlerinnen und Wissenschaftler auf, Forschungsdaten in Fachrepositorien oder in den vorhandenen institutionellen Infrastrukturen der

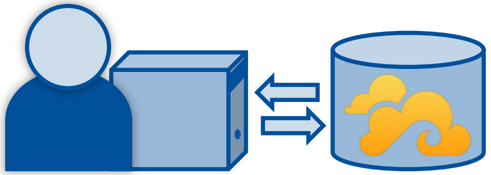
[1] Allianz der deutschen Wissenschaftsorganisationen (2010): Grundsätze zum Umgang mit Forschungsdaten, http://www.allianzinitiative.de/handlungsfelder/forschungsdaten/grundsaetze.html (abgerufen am 28. April 2017).

[2] Deutsche Forschungsgemeinschaft: Leitlinien zum Umgang mit Forschungsdaten (2015). http://www.dfg.de/download/pdf/foerderung/antragstellung/forschungsdaten/richtlinien_forschungsdaten.pdf (abgerufen am 24. April 2017).

[3] Deutsche Forschungsgemeinschaft (2015): Leitlinien zum Umgang mit Forschungsdaten, S. 1. http://www.dfg.de/download/pdf/foerderung/antragstellung/forschungsdaten/richtlinien_forschungsdaten.pdf (abgerufen am 24. April 2017).

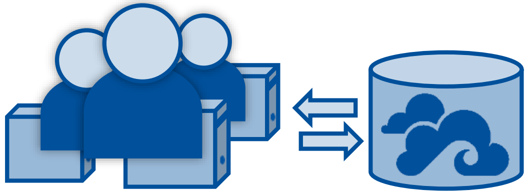
[4] Deutsche Forschungsgemeinschaft: Sicherung guter wissenschaftlicher Praxis (2013), S. 21. http://www.dfg.de/download/pdf/dfg_im_profil/reden_stellungnahmen/download/empfehlung_wiss_praxis_1310.pdf (abgerufen am 24. April 2017).

LUIS services for data storage and backup



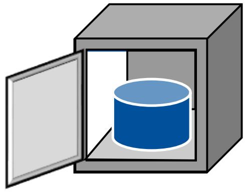
personal “Cloud-Seafile”
(storage + data synchronisation)

go to service description (German)



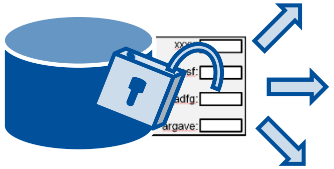
“Projekt-Seafile”
(storage + data synchronisation)

go to service description (German)



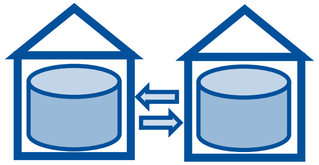
LUH data archive

go to service description (German)



LUH data repository

go to service description (German)



Backup & Restore
(for servers of institutes)

go to service description (German)

How to create re-usable research software

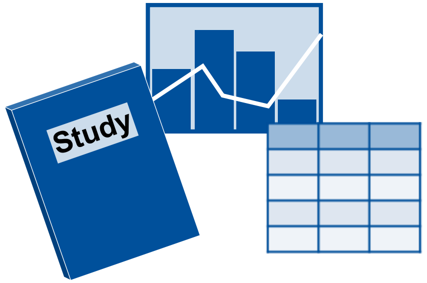
- Documentation of software projects
- The FAIR principles for research software
- Developing a software management plan (SMP)
- Services and infrastructure at LUH
- Foundation: How to set up a project
- Best practices of software projects
    - Tests and Test-driven development (TDD)
    - Refactoring
    - Clean code
    - Error messages
    - README

Documentation of software projects

Data

is not independent from

Research software

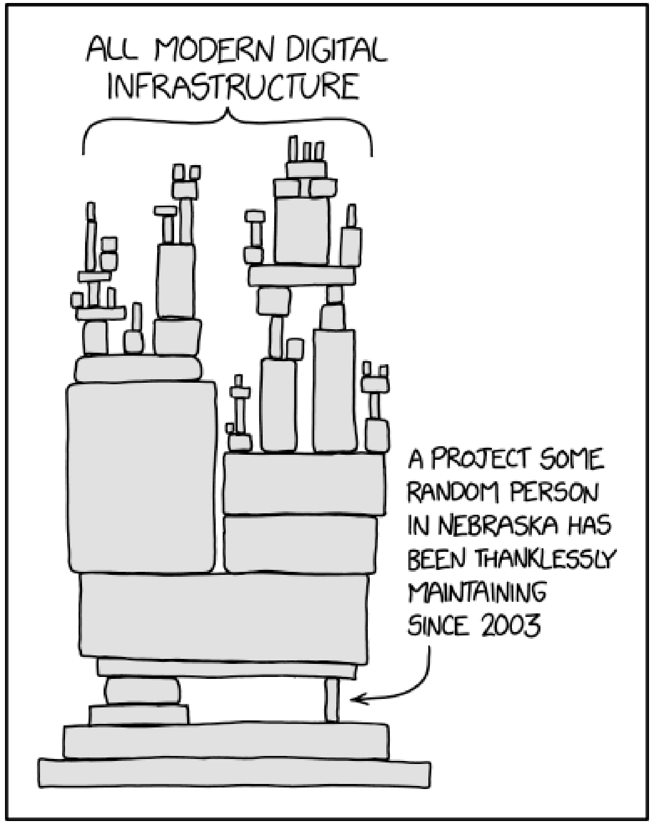


But most researchers

≠

Are no software developers

“Data without software are just numbers”
(Davenport & Grant, 2020)


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

https://xkcd.com/2347

The FAIR Principles for research software

F: Findable

This means that the software and all important related information can be easily found by both humans and machines.

F1. The software, individual important components of the software and the different versions of the software all receive persistent and unique identifiers (PID), e.g. DOI.

F2. The software is described in detail with metadata, which means that all important information for use and subsequent use is included.

F3. Metadata clearly and explicitly include the PID of the software they describe.

F4. Metadata is also FAIR, searchable and can be found and used by metasearch engines.

- ✓ Use persistent identifiers and rich metadata

Findable

RESEARCH DATA ALLIANCE

The FAIR Principles for research software

A: Accessible

Which means that the software and its metadata is retrievable via standardized protocols

A1. For this, the protocol should fulfil some requirements, it should be open and free as well as universally implementable. It should also enable an authentication and authorisation procedure, if necessary.

A2. In addition the metadata are accessible, even when the Software is no longer available.

- ✓ Make software and metadata retrievable

Accessible

The FAIR Principles for research software

I: Interoperable

For software, being interoperable means the possibility to exchange data between independent software. (It differs from data in the way that independent software can not be combined like data to form a new data set.)

I1. When software interacts to exchange data, the exchange protocol need to be clearly described. Domain-relevant standards should be used and APIs should be documented in human and machine readable form.

I2. Software should include references to external data, software or objects, which are required to execute the software. Example: software X is implemented using software A (a programming language)

- ✓ Use APIs, standards and references

Interoperable

The FAIR Principles for research software

R: Reusable

This means that you can not only use the software as it is, but have further information to understand, modify and built new versions upon this software.

R1. The software is described with relevant attributes.

R1.1. A license is given and should be as unrestrictive as possible.

R1.2. Software is associated with detailed provenance: This is a metadata which includes the history of the software, how it came to be and who contributed to it.

R2. Include references to other software which is necessary to compile and run the software.

R3. Your software should meet domain-relevant community standards and coding practices. This can be the choice of programming language, testing methods or file formats.

✓ Document, licence, and follow community standards

Reusable

Developing a software management plan (SMP)

Can help to:

- Create a basis for establishing best practices
- Make the research software reusable and sustainable
- Plan for the necessary resources (financial, human, infrastructure)
- Make it easy to introduce new developers into the project

Ideally an SMP should be drafted at the beginning of a research project. The extent of information in the SMP varies with level of management intensity of the software (low, medium, high).

Software Management Plan (SMP)

A SMP does not need to be long. The following guide offers three templates adapted for different software management levels:

Practical guide to software management plans

The Max Planck Society has published a comprehensive SMP template, which can be integrated in RDMO:

MPG-Template “Software management plan for researcher”

Developing a software management plan (SMP) part I

- Purpose
- Version Control
- Repository
- User Documentation
- Software licensing and compatibility
- Deployment documentation

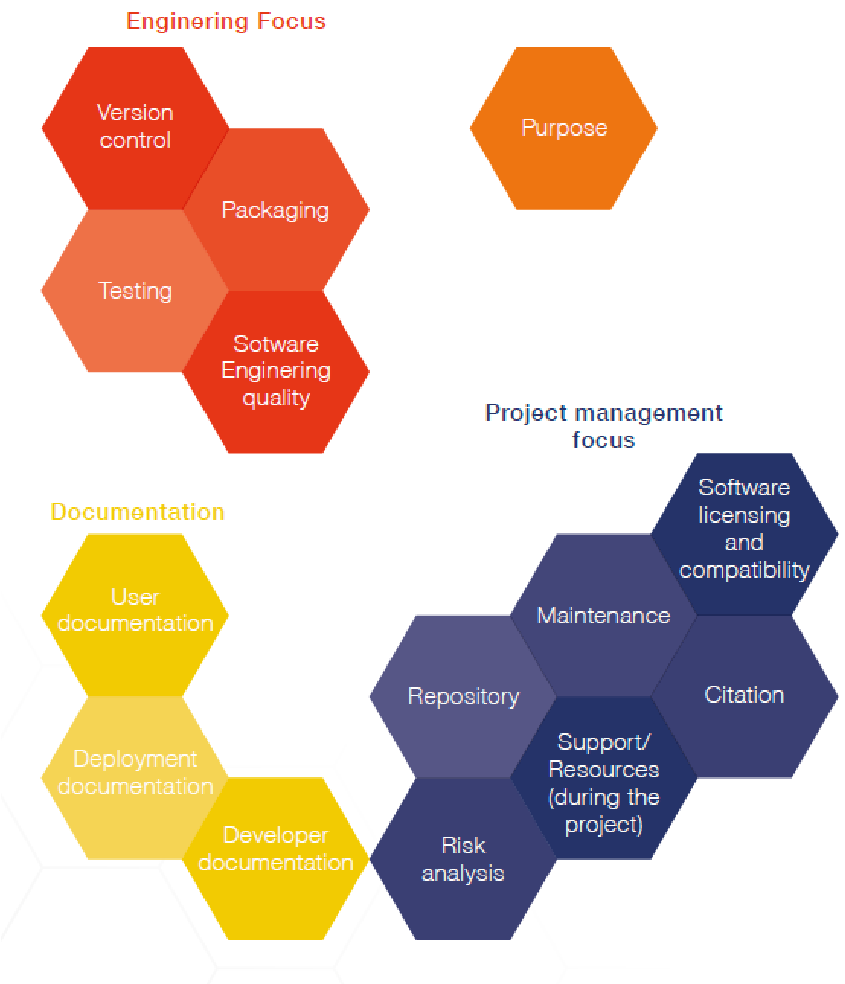
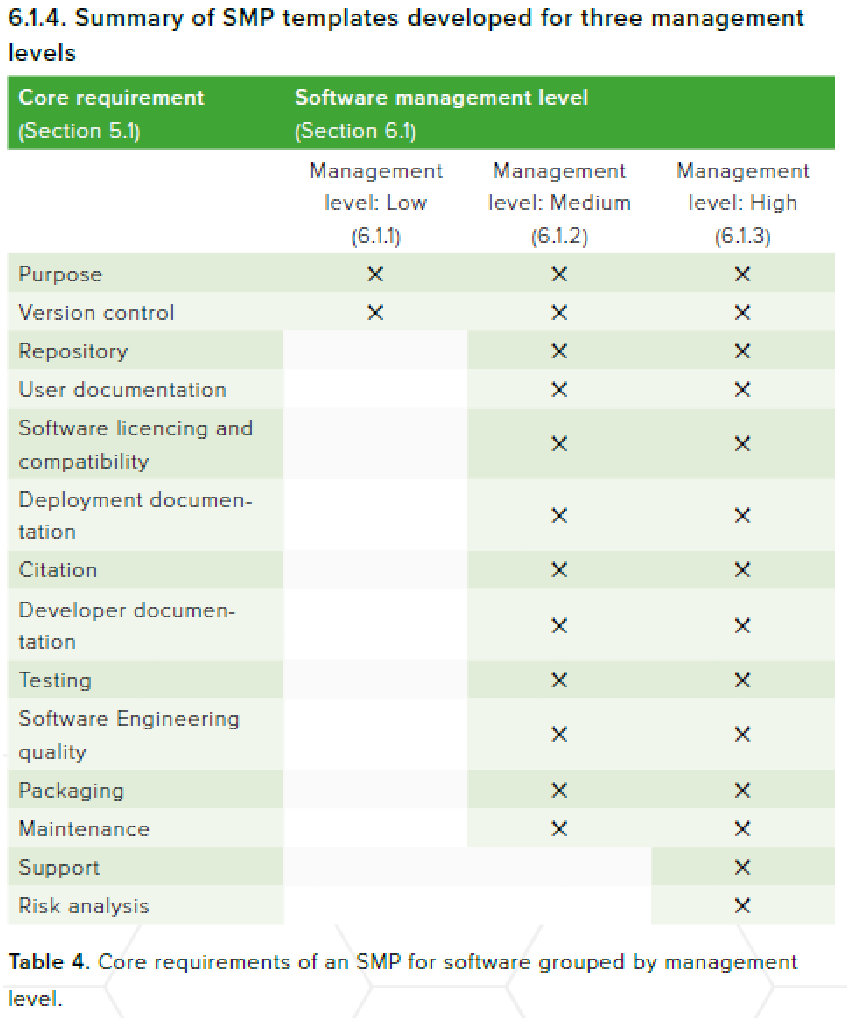


Figure 1. Software Management Plan requirements grouped by their focus.

Graphic from: Martinez-Ortiz, C. et al. (2022), Practical guide to Software Management Plans, p. 17. https://doi.org/10.5281/zenodo.7185371

Developing a software management plan (SMP) part II

- Citation
- Developer documentation
- Testing
- Software Engineering quality
- Packaging
- Maintenance
- Support
- Risk analysis



6.1.4. Summary of SMP templates developed for three management levels

Core requirement (Section 5.1)	**Software management level** (Section 6.1)		
	Management level: Low (6.1.1)	Management level: Medium (6.1.2)	Management level: High (6.1.3)
Purpose	×	×	×
Version control	×	×	×
Repository		×	×
User documentation		×	×
Software licencing and compatibility		×	×
Deployment documentation		×	×
Citation		×	×
Developer documentation		×	×
Testing		×	×
Software Engineering quality		×	×
Packaging		×	×
Maintenance		×	×
Support			×
Risk analysis			×

Table 4. Core requirements of an SMP for software grouped by management level.

Table from: Martinez-Ortiz, C. et al. (2022), Practical guide to Software Management Plans, p.29. https://doi.org/10.5281/zenodo.7185371

Services and infrastructure at LUH

For you

- Version control and software development: GitLab
- Courses for employees and students:
  - Statistics with R
  - Python: Introductory course, Object-oriented programming, Numerical calculation

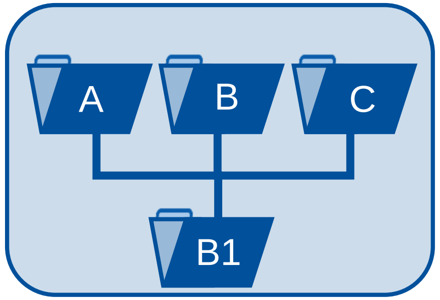
For your institution

- Research cluster housing service: The hardware of an institution can be integrated into the existing cluster system of LUIS
- Scientific Computing: For computationally intensive research project, LUIS provides a cluster system with massively parallel computer systems

Foundation: How to set up a project

Create a similar folder layout:

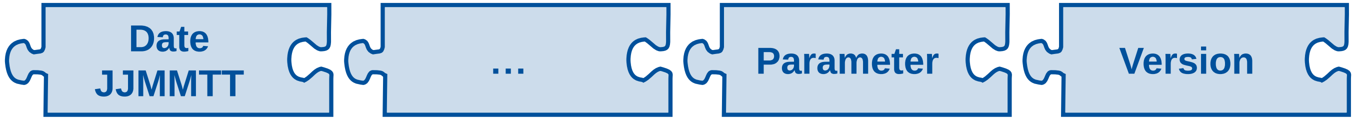
- Licence
- Readme
- Experiments (chronological)
- Provenance
- Scripts



Logical order in subfolders:

- Main script
- Data-generating scripts
- Raw data
- Post-processed data

Modular naming scheme:



Date JJMMTT

…

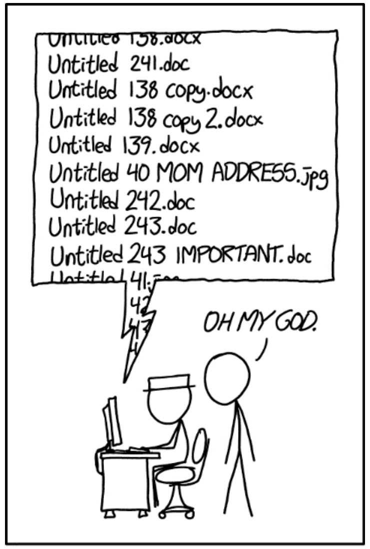
Parameter

Version

Example: 230701_PuppyCuteness_Treats_Cheese_Camembert_v03.fff

Data format: Better not choose “Fancy File Format”

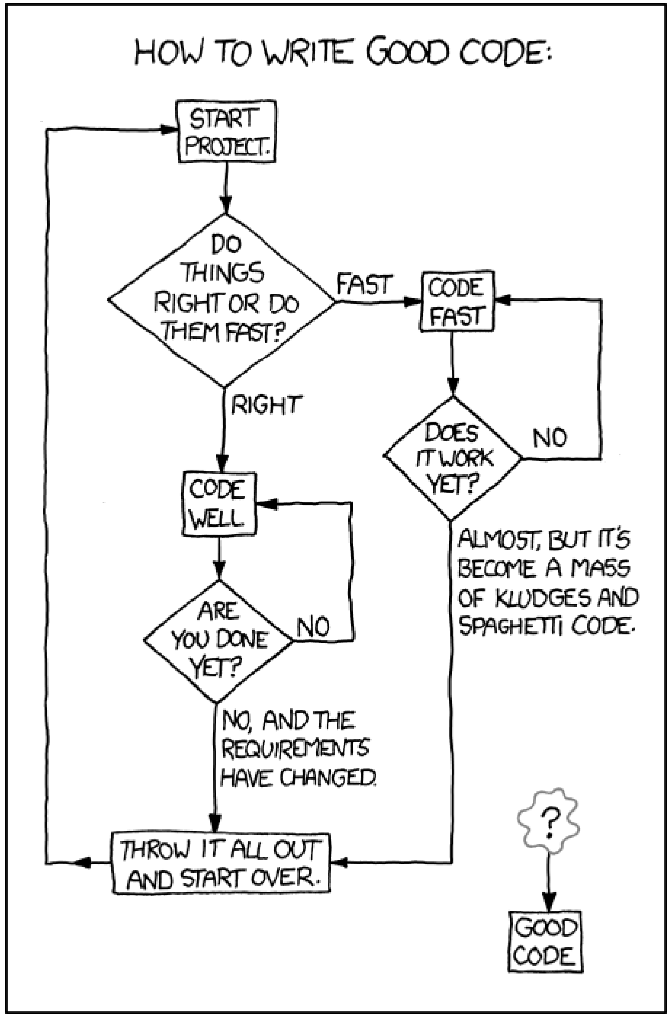
Find a file format



PROTIP: NEVER LOOK IN SOMEONE ELSE'S DOCUMENTS FOLDER.

https://xkcd.com/1459

Best practices of software projects


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

There are a number of published „Best practice“ guides for software development in different scientific fields. We have collected some general practices that help you to write FAIR software.

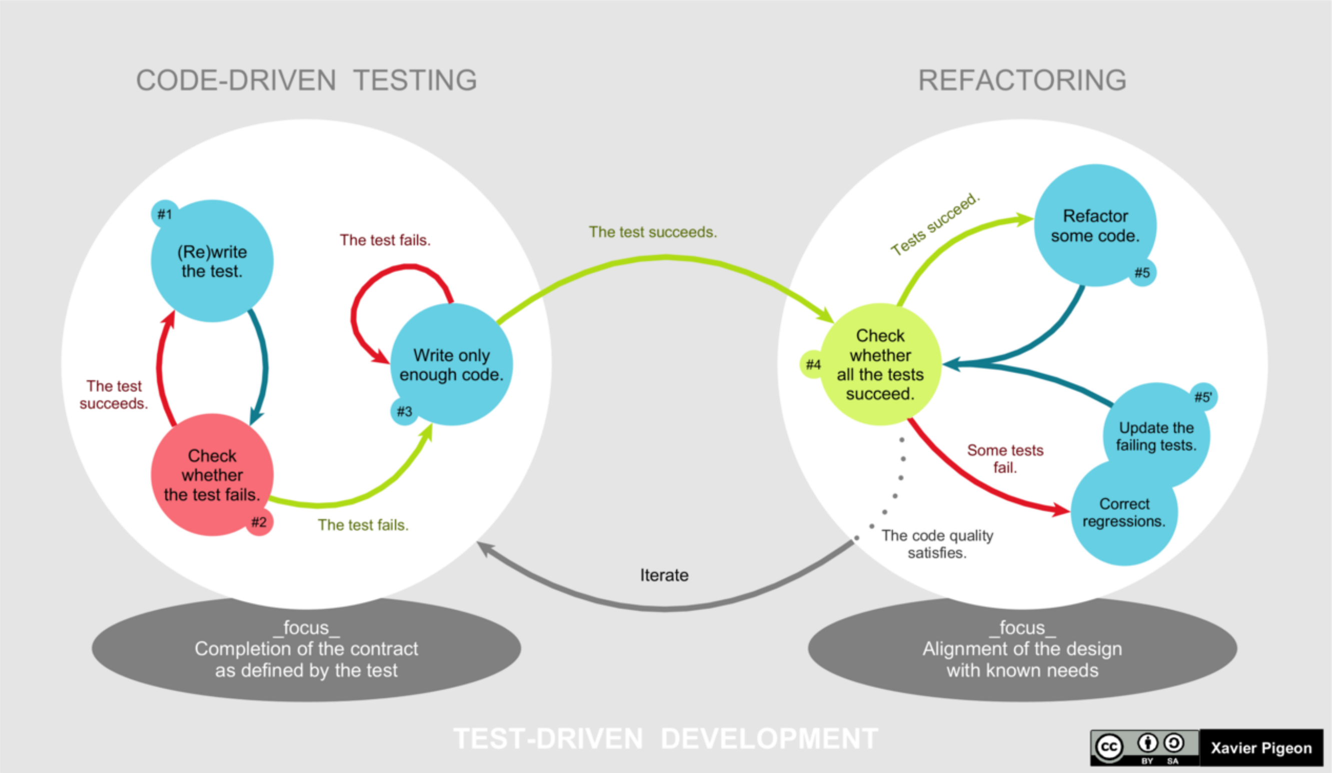
Ten simple rules for documenting scientific software

Oliver Melchert’s best practices for small computational projects

Suresoft
SUSTAINABLE RESEARCH SOFTWARE

Best practice: Tests and Test-driven development (TDD)

- Include testing in the development process
- Each new piece of code is tested for functionality before it is added to the main branch
- Best practice: add incremental changes!
- TDD: First create a test for the desired functionality, then write the code that is able to fulfill the test requirements



Best practice: Refactoring

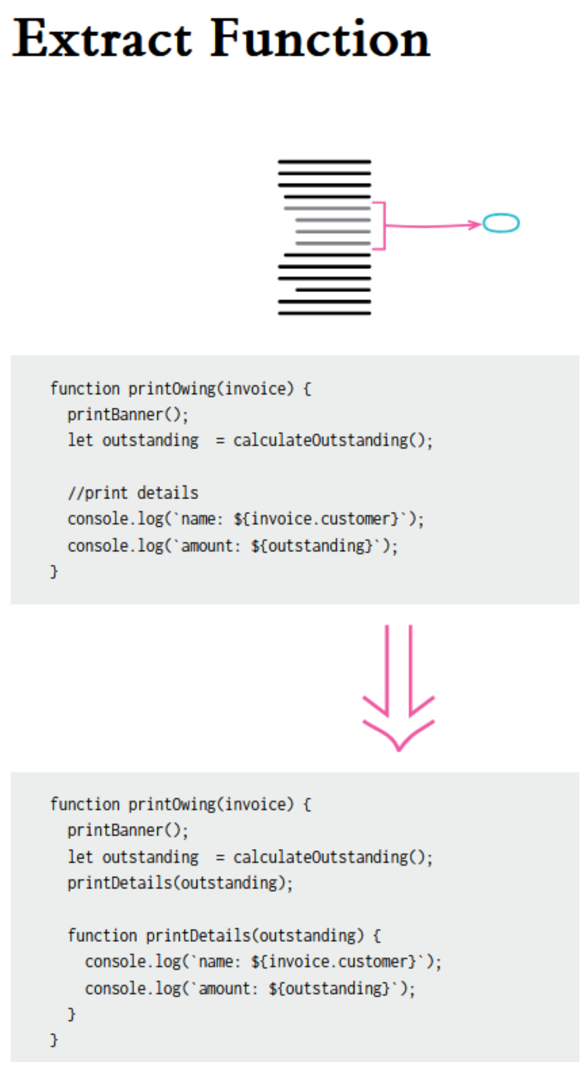
Code refactoring is the process of restructuring code without changing its original functionality.

This helps to ensure that

- the code is more clearly structured and therefore easier to read and understand
- errors and weak points are easier to find and rectify
- the code is easier to extend and better to maintain

Martin Fowler, Refactoring: improving the design of existing code (German)

Robert C. Martin, Clean Code: Refactoring, Patterns, Testen und Techniken für sauberen Code



Extract Function

```
function printOwing(invoice) {
  printBanner();
  let outstanding  = calculateOutstanding();

  //print details
  console.log(`name: ${invoice.customer}`);
  console.log(`amount: ${outstanding}`);
}
```

```
function printOwing(invoice) {
  printBanner();
  let outstanding  = calculateOutstanding();
  printDetails(outstanding);

  function printDetails(outstanding) {
    console.log(`name: ${invoice.customer}`);
    console.log(`amount: ${outstanding}`);
  }
}
```

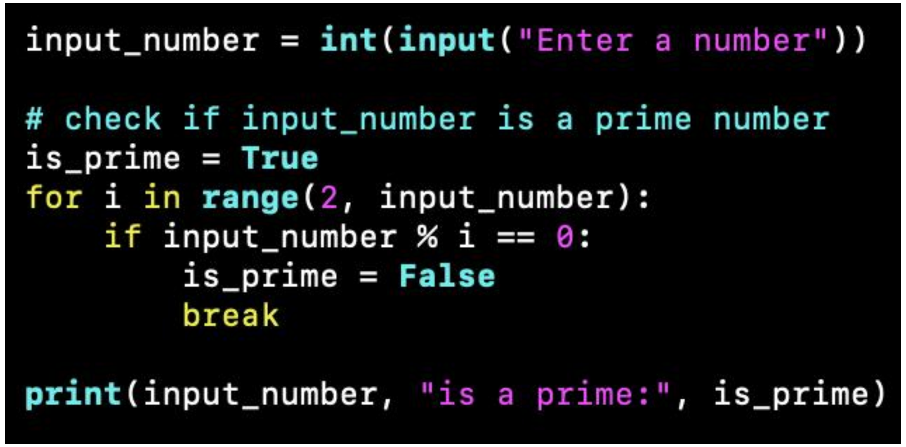
Martin Fowler, Refactoring catalog: https://refactoring.com/catalog/extractFunction.html

Best practice: Clean code

- There are different options how to write names, comments, functions and classes in your code
- Even though a “messy” written code might work, it is not re-usable
- Changing small habits can make a big difference!

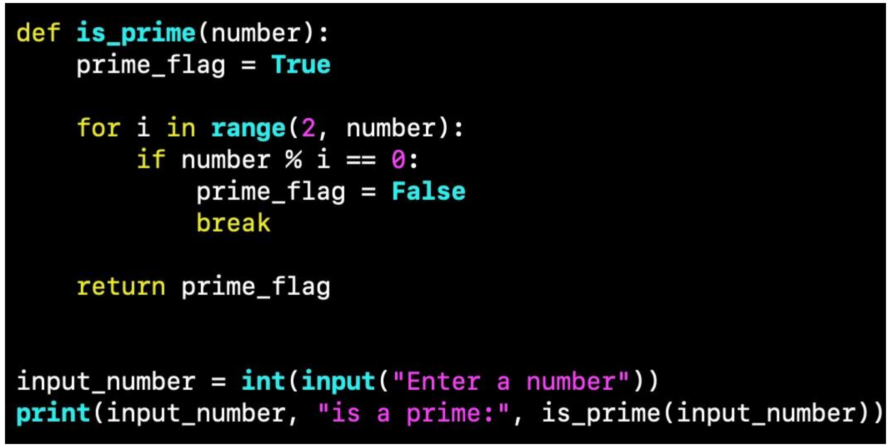
In this example, it is better to define a function which contains the for-loop:

Do not



```python
input_number = int(input("Enter a number"))

# check if input_number is a prime number
is_prime = True
for i in range(2, input_number):
    if input_number % i == 0:
        is_prime = False
        break

print(input_number, "is a prime:", is_prime)
```

Do

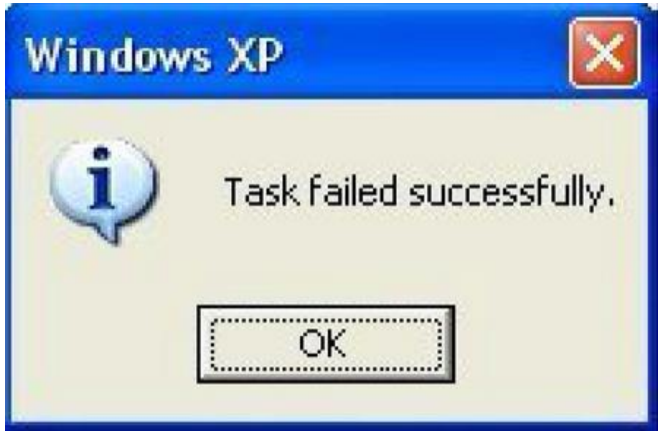


```python
def is_prime(number):
    prime_flag = True

    for i in range(2, number):
        if number % i == 0:
            prime_flag = False
            break

    return prime_flag


input_number = int(input("Enter a number"))
print(input_number, "is a prime:", is_prime(input_number))
```

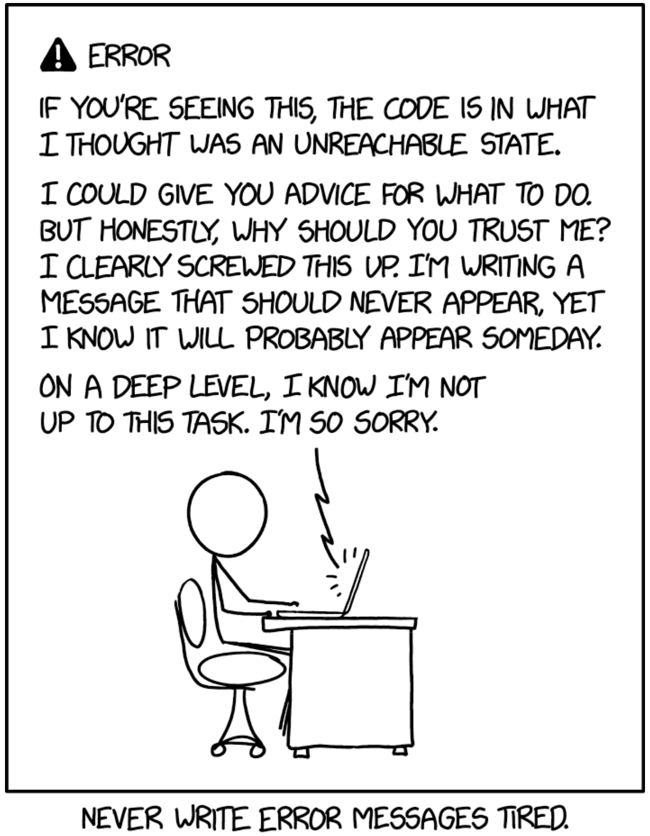
https://suresoft.dev

Best Practice: Error messages

- Write error messages
- Provide solutions or where to find the information relevant to fixing the error.


Windows XP

Task failed successfully.


ERROR

IF YOU'RE SEEING THIS, THE CODE IS IN WHAT I THOUGHT WAS AN UNREACHABLE STATE.

I COULD GIVE YOU ADVICE FOR WHAT TO DO. BUT HONESTLY, WHY SHOULD YOU TRUST ME? I CLEARLY SCREWED THIS UP. I'M WRITING A MESSAGE THAT SHOULD NEVER APPEAR, YET I KNOW IT WILL PROBABLY APPEAR SOMEDAY.

ON A DEEP LEVEL, I KNOW I'M NOT UP TO THIS TASK. I'M SO SORRY.

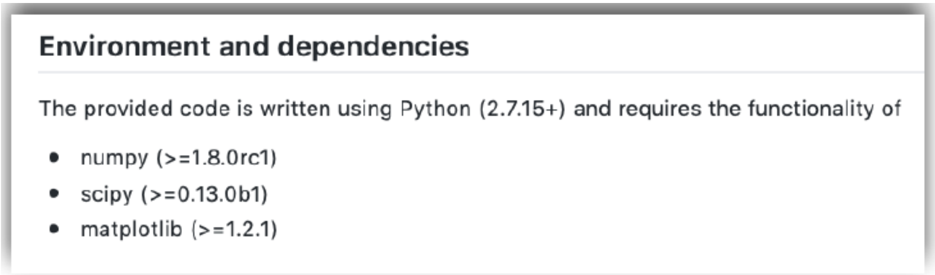
NEVER WRITE ERROR MESSAGES TIRED.

https://xkcd.com/2200

Graphic: Unknown, via: https://community.spiceworks.com/topic/2105369-10-hilarious-error-messages-facepalm-worthy-computer-prompts-that-make-no-sense

Best practice: README Part I

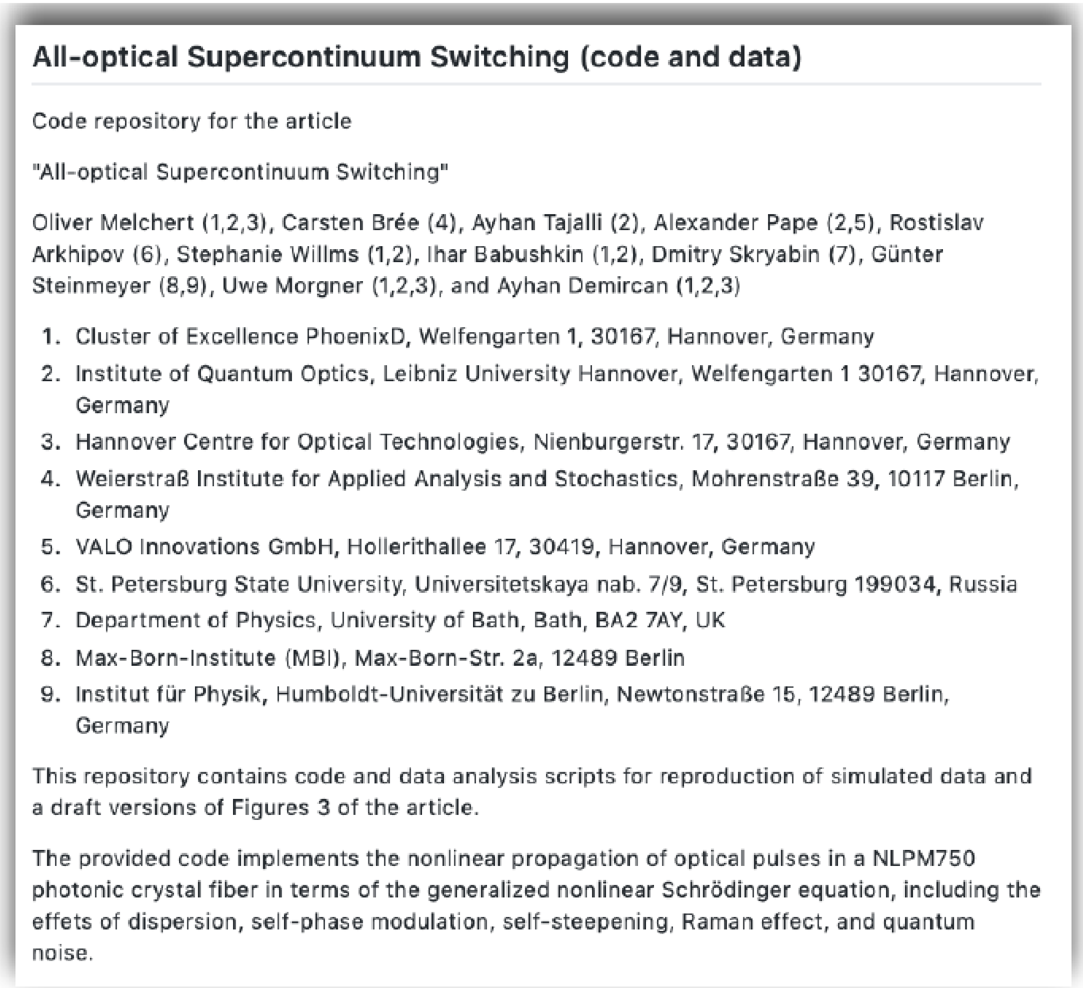
- Give an overview of the project
- List the names and affiliation of people involved
- Environment and dependencies needed to run the code



Environment and dependencies

The provided code is written using Python (2.7.15+) and requires the functionality of

- numpy (>=1.8.0rc1)
- scipy (>=0.13.0b1)
- matplotlib (>=1.2.1)



All-optical Supercontinuum Switching (code and data)

Code repository for the article

"All-optical Supercontinuum Switching"

Oliver Melchert (1,2,3), Carsten Brée (4), Ayhan Tajalli (2), Alexander Pape (2,5), Rostislav Arkhipov (6), Stephanie Willms (1,2), Ihar Babushkin (1,2), Dmitry Skryabin (7), Günter Steinmeyer (8,9), Uwe Morgner (1,2,3), and Ayhan Demircan (1,2,3)

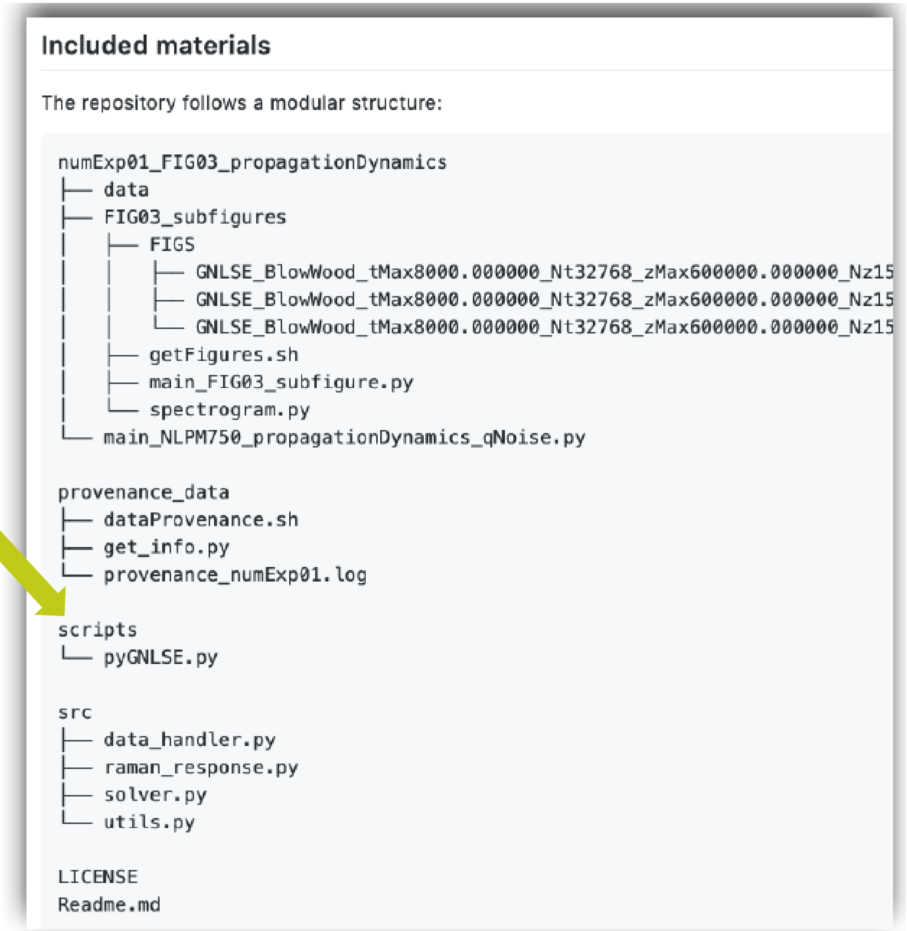
1. Cluster of Excellence PhoenixD, Welfengarten 1, 30167, Hannover, Germany
2. Institute of Quantum Optics, Leibniz University Hannover, Welfengarten 1 30167, Hannover, Germany
3. Hannover Centre for Optical Technologies, Nienburgerstr. 17, 30167, Hannover, Germany
4. Weierstraß Institute for Applied Analysis and Stochastics, Mohrenstraße 39, 10117 Berlin, Germany
5. VALO Innovations GmbH, Hollerithallee 17, 30419, Hannover, Germany
6. St. Petersburg State University, Universitetskaya nab. 7/9, St. Petersburg 199034, Russia
7. Department of Physics, University of Bath, Bath, BA2 7AY, UK
8. Max-Born-Institute (MBI), Max-Born-Str. 2a, 12489 Berlin
9. Institut für Physik, Humboldt-Universität zu Berlin, Newtonstraße 15, 12489 Berlin, Germany

This repository contains code and data analysis scripts for reproduction of simulated data and a draft versions of Figures 3 of the article.

The provided code implements the nonlinear propagation of optical pulses in a NLPM750 photonic crystal fiber in terms of the generalized nonlinear Schrödinger equation, including the effets of dispersion, self-phase modulation, self-steepening, Raman effect, and quantum noise.

Best practice: README Part II

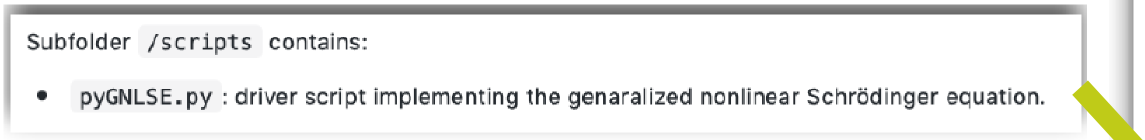
- Details about included material
- License, Citation & Acknowledgements



Included materials

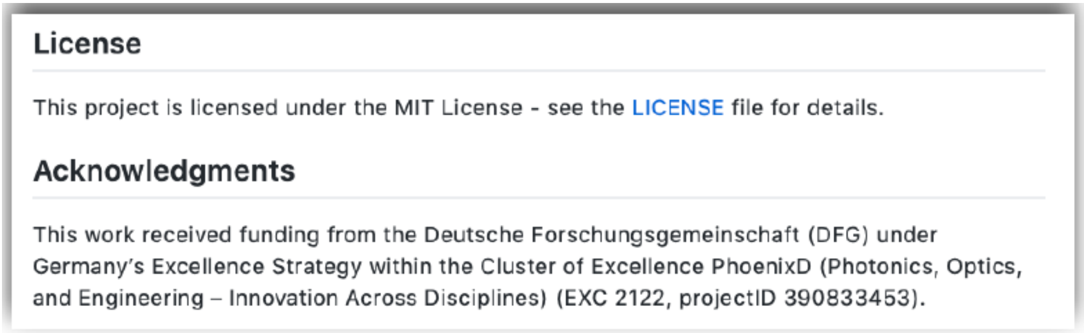
The repository follows a modular structure:

```
numExp01_FIG03_propagationDynamics
├── data
├── FIG03_subfigures
│   ├── FIGS
│   │   ├── GNLSE_BlowWood_tMax8000.000000_Nt32768_zMax600000.000000_Nz15
│   │   ├── GNLSE_BlowWood_tMax8000.000000_Nt32768_zMax600000.000000_Nz15
│   │   └── GNLSE_BlowWood_tMax8000.000000_Nt32768_zMax600000.000000_Nz15
│   ├── getFigures.sh
│   ├── main_FIG03_subfigure.py
│   └── spectrogram.py
└── main_NLPM750_propagationDynamics_qNoise.py

provenance_data
├── dataProvenance.sh
├── get_info.py
└── provenance_numExp01.log

scripts
└── pyGNLSE.py

src
├── data_handler.py
├── raman_response.py
├── solver.py
└── utils.py

LICENSE
Readme.md
```



Subfolder `/scripts` contains:

- `pyGNLSE.py` : driver script implementing the genaralized nonlinear Schrödinger equation.



License

This project is licensed under the MIT License - see the LICENSE file for details.

Acknowledgments

This work received funding from the Deutsche Forschungsgemeinschaft (DFG) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (Photonics, Optics, and Engineering – Innovation Across Disciplines) (EXC 2122, projectID 390833453).

How to publish research software and code

- Why publish your data and software?
- Preparation of your software publication
    - Version control your documentation
    - Use automated documentation tools
    - Software citation
    - User guide and examples
- Publishing software
- Licences
- Linking journal articles with related data and software
- Example: Publish software via Zenodo

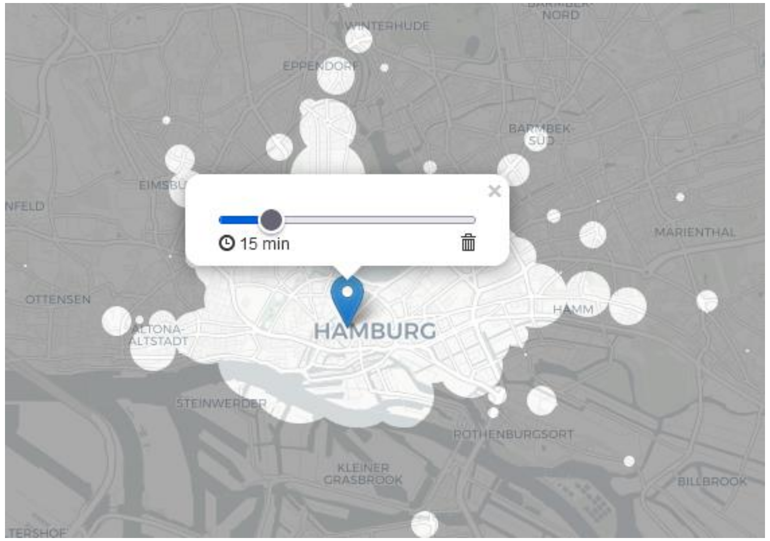
Why publish your data and software?

Open data are valuable and can be used to create new research output or tools of public interest:

- Participation: findtoilet.dk
- Empowerment of citizens: mapnificient
- Innovation: EHEC outbreak genome analysis

Reasons to publish your research software:

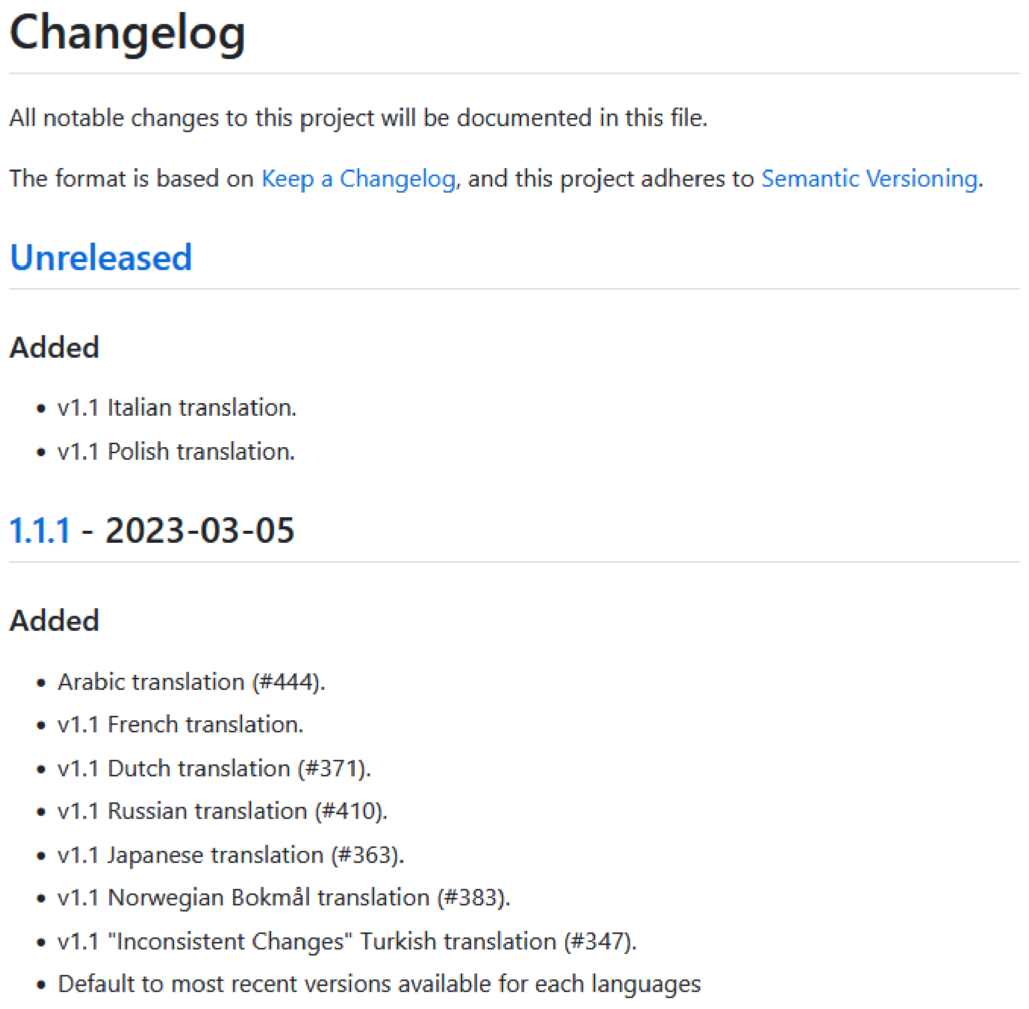
- To enable software citation
- To make the software FAIR
- To let software count towards evaluation



Mapnificent.net

Software publication: Version control your documentation

- Make it clear which documentation applies to which software version
- A changelog can make this task much easier
- There are useful services that can help you do this.



Changelog

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog, and this project adheres to Semantic Versioning.

Unreleased

Added

- v1.1 Italian translation.
- v1.1 Polish translation.

1.1.1 - 2023-03-05

Added

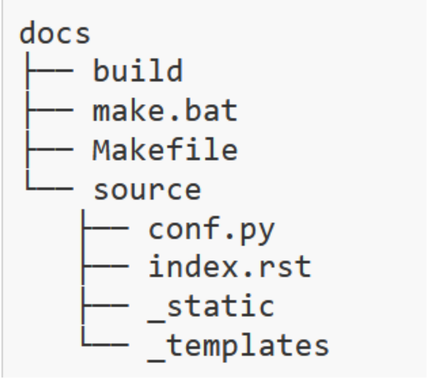
- Arabic translation (#444).
- v1.1 French translation.
- v1.1 Dutch translation (#371).
- v1.1 Russian translation (#410).
- v1.1 Japanese translation (#363).
- v1.1 Norwegian Bokmål translation (#383).
- v1.1 "Inconsistent Changes" Turkish translation (#347).
- Default to most recent versions available for each languages

Screenshot: https://github.com/olivierlacan/keep-a-changelog/blob/main/CHANGELOG.md

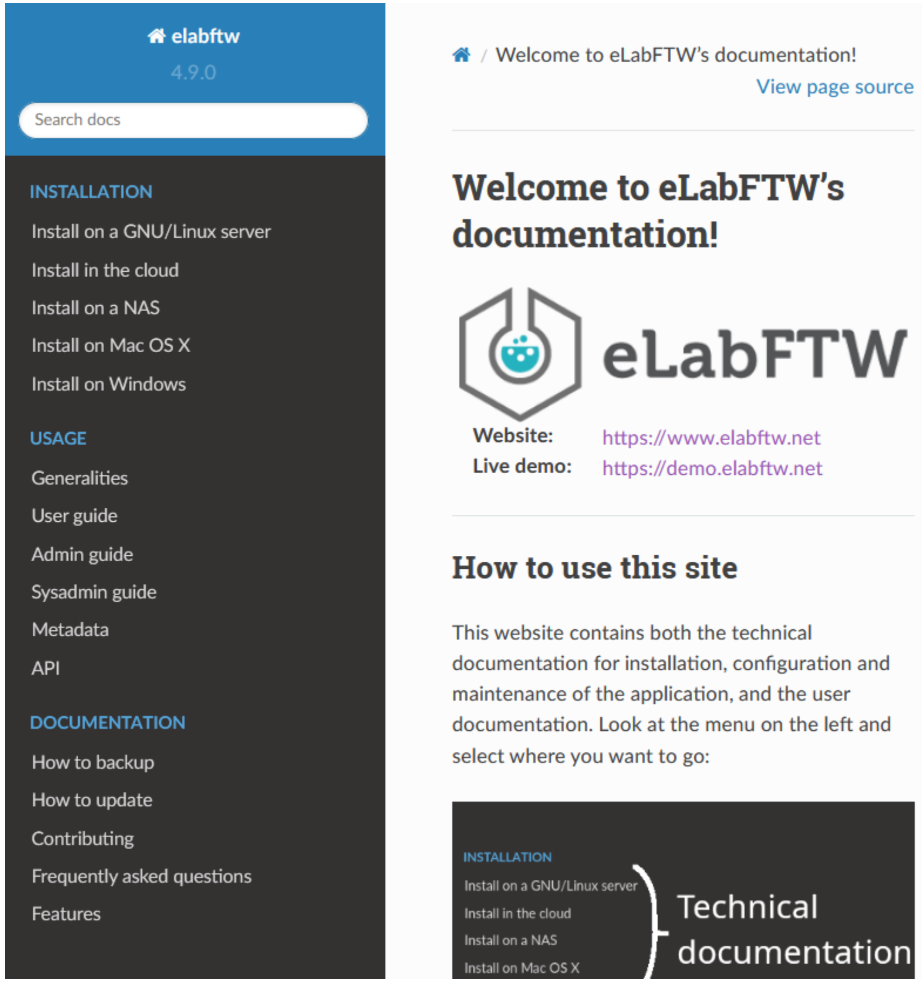
Best Practice: Use documentation tools

- Documentation of a project is the alpha and omega
- Complete and prompt documentation is essential
- Use tools to create a nice documentation site which helps others to use your software

Sphinx Python Documentation Generator:



```
docs
├── build
├── make.bat
├── Makefile
└── source
    ├── conf.py
    ├── index.rst
    ├── _static
    └── _templates
```



Welcome to eLabFTW's documentation!

Website: https://www.elabftw.net

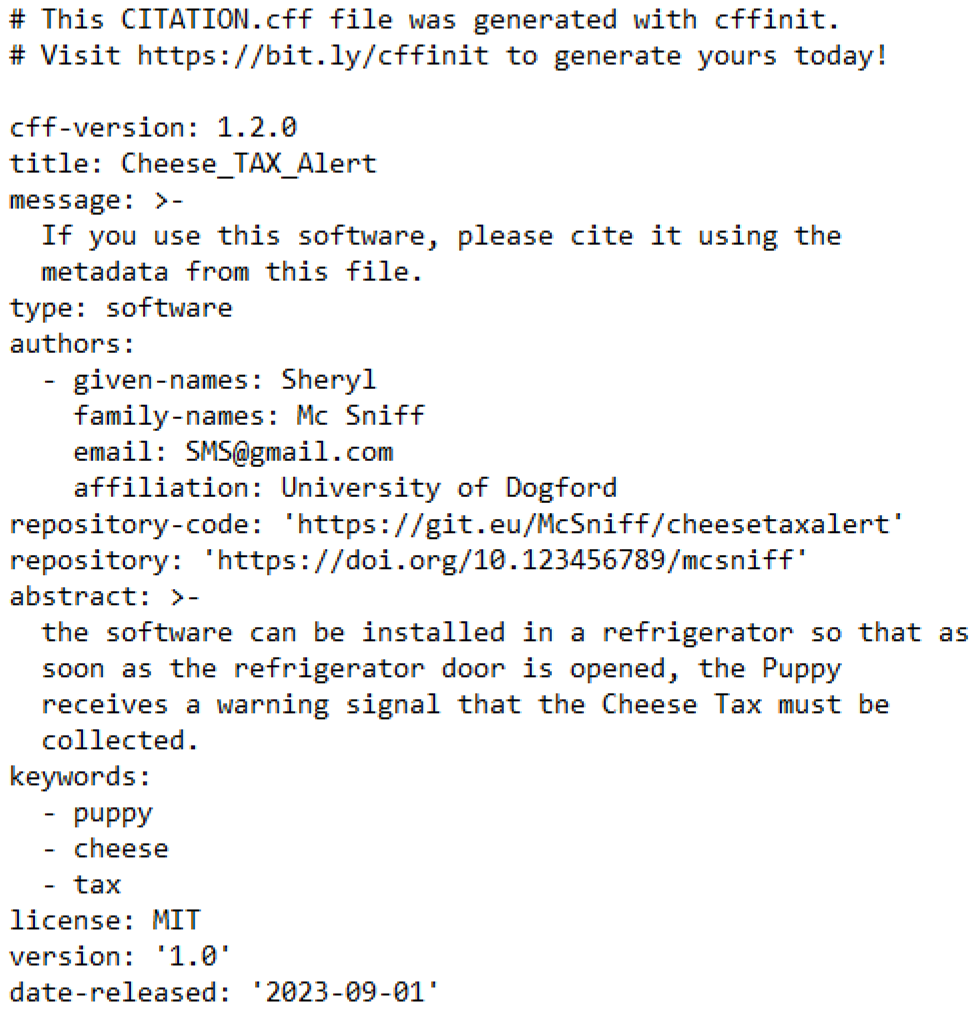
Live demo: https://demo.elabftw.net

How to use this site

This website contains both the technical documentation for installation, configuration and maintenance of the application, and the user documentation. Look at the menu on the left and select where you want to go:

Software publication: Software Citation

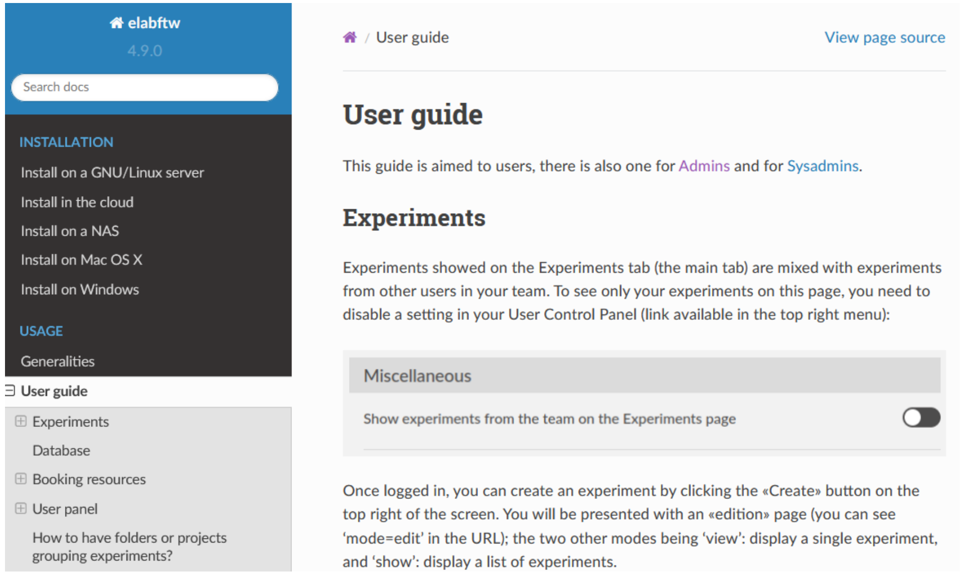
- Provide information on how to cite your code
- Be sure to always properly cite other people's code that you reuse as well



```
# This CITATION.cff file was generated with cffinit.
# Visit https://bit.ly/cffinit to generate yours today!

cff-version: 1.2.0
title: Cheese_TAX_Alert
message: >-
  If you use this software, please cite it using the
  metadata from this file.
type: software
authors:
  - given-names: Sheryl
    family-names: Mc Sniff
    email: SMS@gmail.com
    affiliation: University of Dogford
repository-code: 'https://git.eu/McSniff/cheesetaxalert'
repository: 'https://doi.org/10.123456789/mcsniff'
abstract: >-
  the software can be installed in a refrigerator so that as
  soon as the refrigerator door is opened, the Puppy
  receives a warning signal that the Cheese Tax must be
  collected.
keywords:
  - puppy
  - cheese
  - tax
license: MIT
version: '1.0'
date-released: '2023-09-01'
```

Software publication: User guide and examples

- ➢ People might not use your software if it takes too much time to get started
- ➢ Add examples, a tutorial, videos – anything that helps to show the functionalities of your software
- ➢ If you have numerous examples, use a special section or directory
- ➢ Try your user guide on someone who does not know your software yet


elabftw
4.9.0
Search docs
INSTALLATION
Install on a GNU/Linux server
Install in the cloud
Install on a NAS
Install on Mac OS X
Install on Windows
USAGE
Generalities
User guide
Experiments
Database
Booking resources
User panel
How to have folders or projects grouping experiments?

/ User guide   View page source

User guide

This guide is aimed to users, there is also one for Admins and for Sysadmins.

Experiments

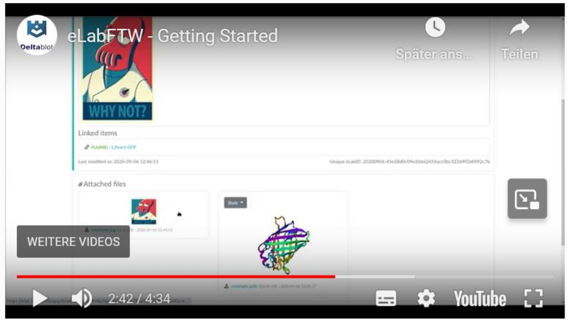
Experiments showed on the Experiments tab (the main tab) are mixed with experiments from other users in your team. To see only your experiments on this page, you need to disable a setting in your User Control Panel (link available in the top right menu):

Miscellaneous

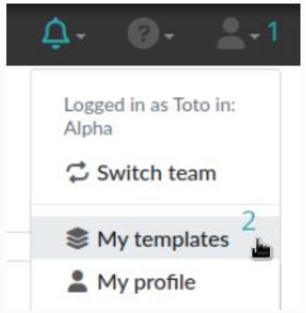
Show experiments from the team on the Experiments page

Once logged in, you can create an experiment by clicking the «Create» button on the top right of the screen. You will be presented with an «edition» page (you can see 'mode=edit' in the URL); the two other modes being 'view': display a single experiment, and 'show': display a list of experiments.

Video tutorials



Screenshots



Publishing software

- Publish software in a publication repository (like Zenodo or your institutional data repository)
    - This gives you:
        - An unique and persistent identifier (e.g. DOI)
        - An identifier that points to the specific version of your software
- Provide citation metadata for your software (via CFF)
- License your code
- Possibility of additional publication in a software journal

Repository publication → software journal → papers

Important: The source code repository (e.g. GitHub, GitLab) is not a publication repository!

Licenses

Common software licenses:

- MIT
- MPL 2.0
- GPL 3.0 – only

Choose an open source license

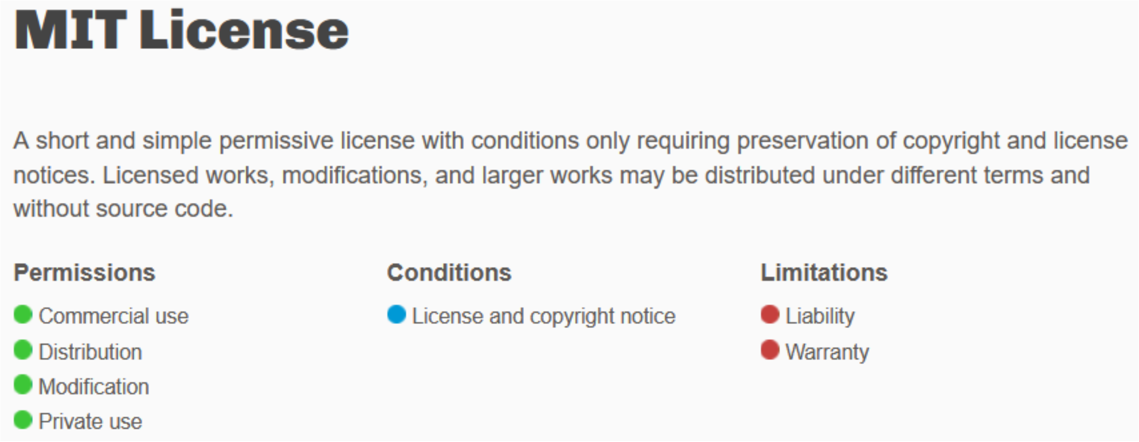
Licenses for text and data:

Creative Commons licenses

- CC0 – no name attribution, no restrictions
- CC-BY – author attribution when using the data
- CC-BY-SA – author attribution and share under the same conditions

Choose a CC license

Example: Overview of the MIT-License



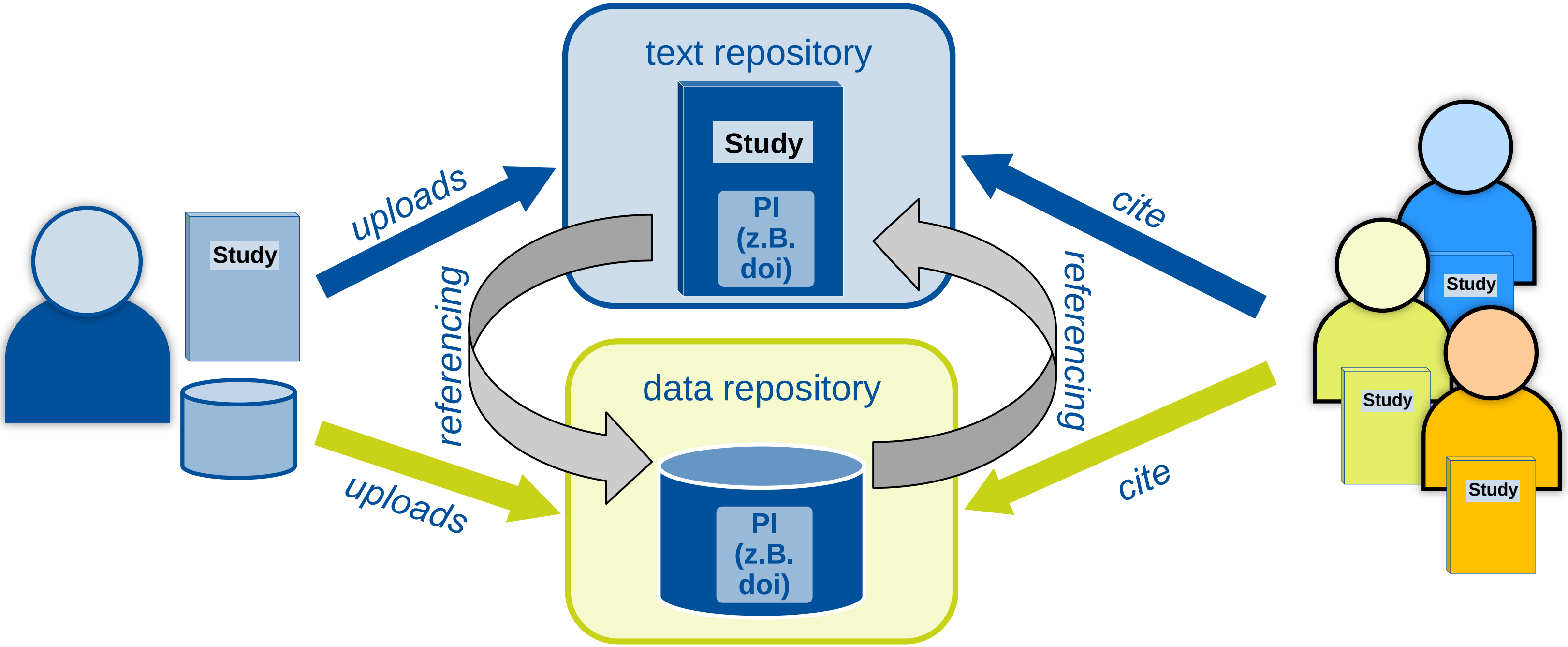
MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions	Conditions	Limitations
🟢 Commercial use	🔵 License and copyright notice	🔴 Liability
🟢 Distribution		🔴 Warranty
🟢 Modification		
🟢 Private use		

https://choosealicense.com/licenses/mit/

Linking journal articles with related data and software



further reading

Diana Kwon (2019): The Push to Replace Journal Supplements with Repositories. Online article, 19 August 2019, on the website of „The Scientist“.

go to article

i

LUH also provides its own institutional text repository for open access publications.

go to text repository

Summary

- Software management happens before you even start coding:
  - Clarify responsibilities
  - Agree on standards and methods and write them down (SMP)
  - Recognize that data and software management are work tasks that will consume some of your time
- Develop a culture that values data- and software management
  - Include the best practices in your everyday work
  - Think of your future self and anyone who might want to re-use your code
  - Engage in projects of others: Create and solve issues, reuse instead of setting up new

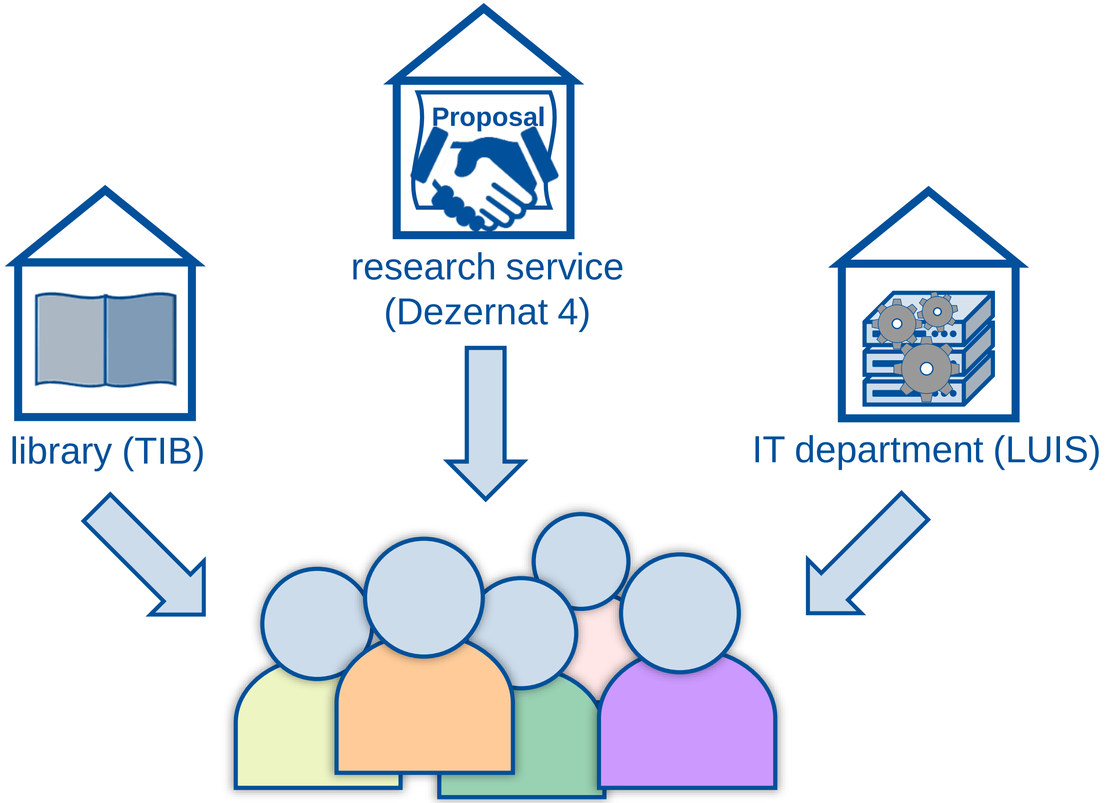
#DEAR FUTURE SELF,
#
# YOU'RE LOOKING AT THIS FILE BECAUSE
# THE PARSE FUNCTION FINALLY BROKE.
#
# IT'S NOT FIXABLE. YOU HAVE TO REWRITE IT.
# SINCERELY, PAST SELF
DEAR PAST SELF, IT'S KINDA CREEPY HOW YOU DO THAT.
#ALSO, IT'S PROBABLY AT LEAST
# 2013. DID YOU EVER TAKE
#THAT TRIP TO ICELAND?
STOP JUDGING ME!

https://xkcd.com/1421/

Supporting services and initiatives

- The Research Data Support Team of LUH
- External information and support
- RSE working groups and initiatives

The Research Data Support Team of LUH



Sign up for our monthly newsletter on RDM!

i

We train and counsel LUH members on the following topics, among others:

- documentation and publication of data
- DMP and RDM policies
- RDM statements in funding proposals
- LUH services and infrastructures for data management
- legal issues (in cooperation with the legal department and the data protection office)
- practical implementation of data management in research processes

Please note as well the additional information, guides and training courses on our website.

go to website of the Research Data Support Team

External information and support

Courses, talks and schools:

Suresoft Project: Workshop schedule with courses on best practices in research software engineering

Hessian Research Data Infrastructures: Data and Code Schools, Data Talks

Helmholtz Information & Data Science Academy:
Courses, schools, workshops & talks

Self-training courses:

Suresoft Knowledge Hub: Workshops on Git foundations, Clean Code, Software Testing, CI and more

Software Carpentry: You can work through the lessons on the Unix Shell, Git, Programming with Python or R by yourself

RSE working groups and initiatives

Research Software Engineers (RSEs)

- German association (de-RSE e.V.)
- monthly call, Chat, Blog
- International conferences for and by Research Software Engineers

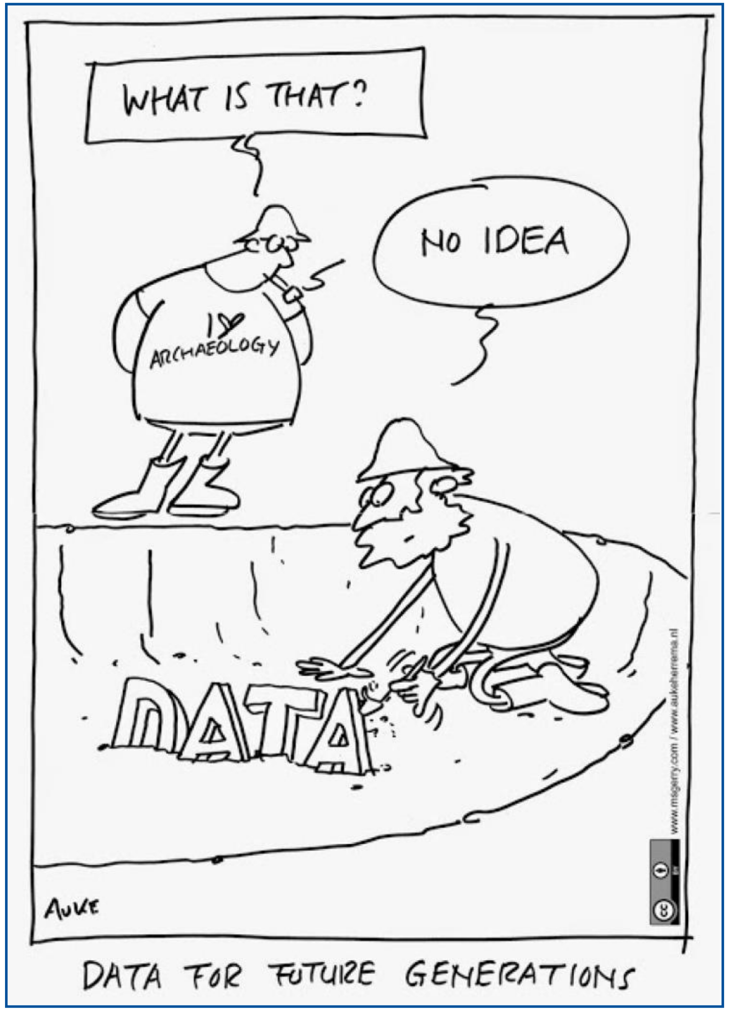
Software Sustainability Institute (UK)

- Motto: “Better software, better research”
- Research Software Healthcheck: Evaluation tool to check your software
- online Research Software Camps

Research Data Alliance (RDA): International network of RDM specialists with a special focus on technical aspects.

NFDI: The federal government and the state governments jointly fund the development of a national research data infrastructure (NFDI). X consortia are developing disciplinary standards, establish services and offer trainings.

Thank you for taking this course!


WHAT IS THAT?
NO IDEA
I ♥ ARCHAEOLOGY
DATA
www.misgery.com / www.aukeherrema.nl
AUKE
DATA FOR FUTURE GENERATIONS

by Auke Herrema