February 8, 2024 | 1:30-3pm
Virtual Workshop

Best Practices for Managing Your Code & Scripts You Use to Generate Your Research

u.mcmaster.ca/scds-events

SCDS

Library | McMaster University

Best Practices for Managing Your Code and Scripts to Generate Your Research

Research Software Development

- Kelvin Lee
- Ola Hejazi
- David Beardwood

Research Data Management

- Emilie Altman
- Danica Evering

McMaster University is located on the traditional Territories of the Mississauga and Haudenosaunee Nations, and within the lands protected by the “Dish With One Spoon” wampum agreement.

Laslovarga, “Webster Falls in Winter, Waterdown, Hamilton, Ontario, Canada - Spencer Gorge / Webster's Falls Conservation Area,” 23 January 2011, Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Waterdawn_Webster_Falls_in_Winter8.jpg

Code of Conduct

The Sherman Centre and the McMaster University Library are committed to fostering a supportive and inclusive environment for its presenters and participants.

As a participant in this session, you agree to support and help cultivate an experience that is collaborative, respectful, and inclusive, as well as free of harassment, discrimination, and oppression. We reserve the right to remove participants who exhibit harassing, malicious, or persistently disruptive behaviour.

Please refer to our code of conduct webpage for more information: scds.ca/events/code-of-conduct/

Session Recording and Privacy

This session is being recorded with the intention of being shared publicly via the web for future audiences. In respect of your privacy, participant lists will not be shared outside of this session, nor will question or chat transcripts.

Questions asked via the chat box will be read by the facilitator without identifying you. Note that you may be identifiable when asking a question during the session in an audio or visual format.

Certificate Program

The Sherman Centre offers a Certificate of Attendance that rewards synchronous participation at 7 workshops. We also offer concentrations in Data Analysis and Visualization, Digital Scholarship, and Research Data Management.

Learn more about the Certificate Program: https://scds.ca/certificate-program
Verify your participation at a session: https://u.mcmaster.ca/verification
At an unspecified point during the workshop, a code will be read aloud. This is the answer to the third question of the form.

Reproducible Research

FAIR Principles

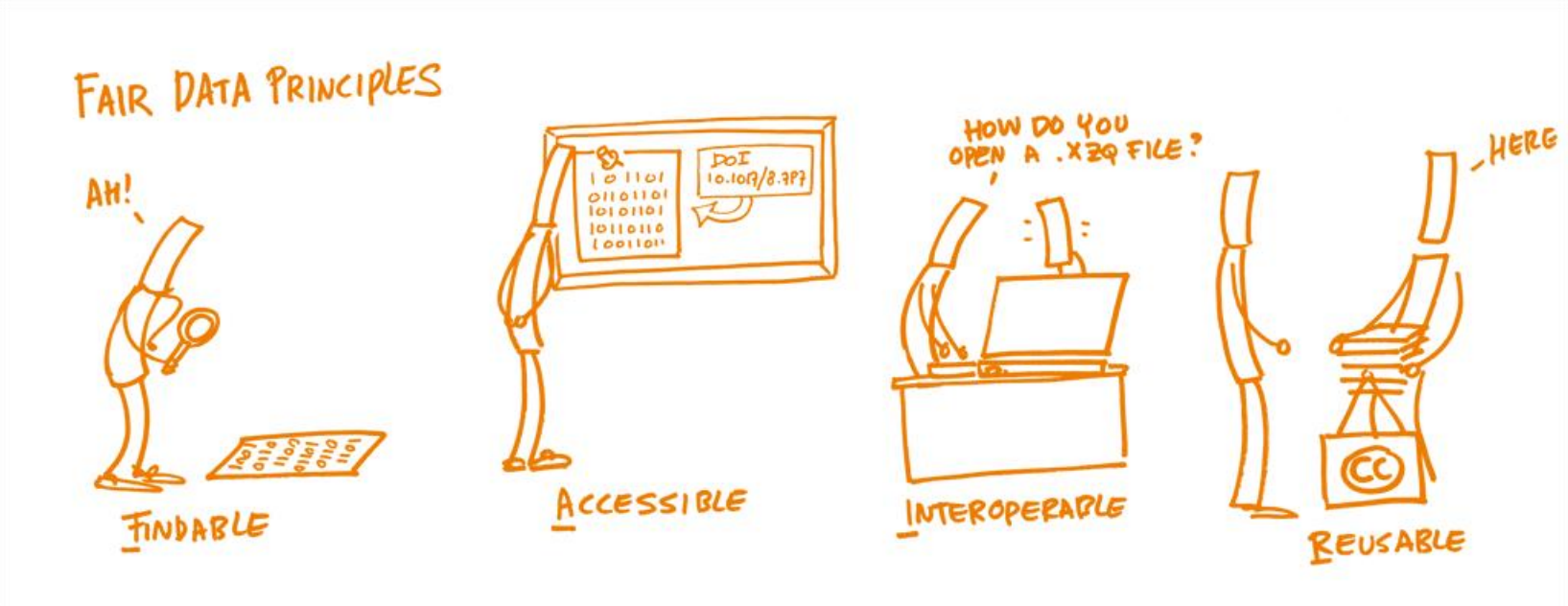
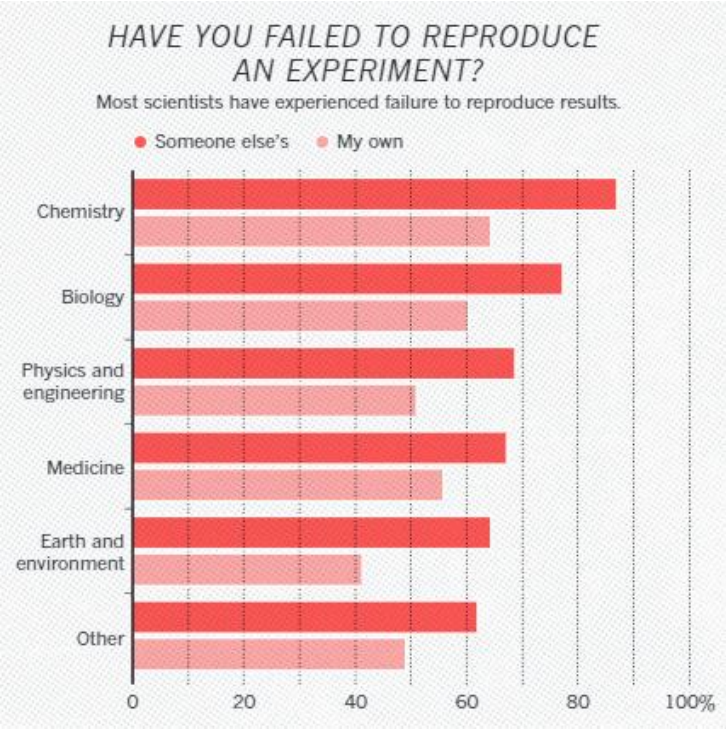


Image retrieved from *fosteropenscience.eu*.

Reproducible Research

Open Research and Reproducibility

- Reproducibility is essential for research verification and integrity
- **The Reproducibility Crisis**
  - In recent years, many researchers have failed to replicate previous scientific experiments
  - The Replication Project replicated 100 prominent psychology studies and succeeded in only 39% of attempts[1]
  - Baker surveyed 1500 scientists – high replication failure rates[2]
  - While contextual reasons may be a factor, it makes clear the importance of replication in science



[1] Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. Science, 349(6251), aac4716. Doi: 10.1126/science.aac4716

[2] Baker M .1,500 scientists lift the lid on reproducibility. NatureNews. 2016; 533:452

Reproducible Research

Open Research and Reproducibility

- How can we make our research reproducible?

1. **Open Data**
   - Increases the reliability of the study – replicable
   - Increases opportunities for collaboration
   - Increases citations
   - Often required by journals and funding agencies
2. **Documentation**
   - Good documentation is needed for data to be useful!
   - Will you understand your own data in 5 years? What about someone who has never seen your data before?

Reproducible Research

Data Management Plans are an integral part of good research data management practices. This lays out your plan to create, store, organize, document, secure, preserve, and share your research data. It's an essential part of the research data lifecycle and good data management practices.

Preservation

- FAIR principles
- Persistent identifiers
- **Data repositories**
- Open file formats
- Metadata

Planning

- **Data Management Plans**
- Grant applications
- Ethics applications
- Data sharing agreements
- Data access

Publication

- **Data sharing**
- Code sharing

Data Collection

- Electronic Lab Notebooks
- De-identification
- Data documentation
- File management
- Data storage
- Data security

Analysis

Reproducible Research

Data Management Plan (DMP)

- A tool to help you plan to create, store, organize, document, secure, preserve, and share your research data.
- A **living document –** something you'll work with, adapt, and change through your research.
- Create it at the start of your research - avoid pitfalls and problems before they occur.
- Prepare for future stages of research including potential data sharing (if desired) and preservation.
- Research is a team effort – collaborate on your DMP.
- Many research funders require grant applicants to submit a DMP – including the Tri-Agencies (NSERC, CIHR, SSHRC – started 2022), NIH, and others.

Reproducible Research Software

Best Practices for Managing Your Code and Scripts You Use to Generate Your Research

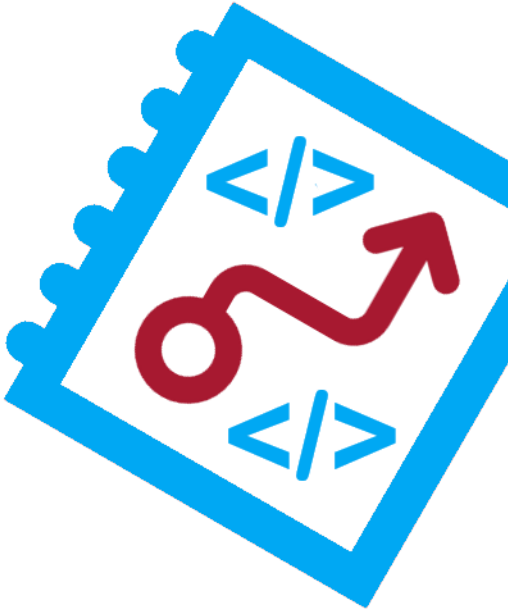
through

Software Management Plans

Reproducible Research Software

~~Data~~ Software Management Plan (~~DMP~~ SMP)

- A tool to help you plan to create, store, organize, document, secure, preserve, and share your research ~~data~~ software.
- A **living document** – something you'll work with, adapt, and change through your research.
- Create it at the start of your research - avoid pitfalls and problems before they occur.
- Prepare for future stages of research including potential data sharing (if desired) and preservation.
- Research is a team effort – collaborate on your SMP.
- Many research funders are evaluating SMP requirements – Tri-Agency Research Data Management Policy: deposit alongside research data any "code that directly support the research conclusions in journal publications and pre-prints that arise from agency-supported research."



Reproducible Research Software

Why include software in my management plans?

- Researchers increasingly rely on software in their research.
- A survey by the Software Sustainability Institute, carried out among UK researchers, found that 92% of academics use research software, and 7 out of 10 researchers deemed it impossible to conduct their research without it.[1]

That's a lot of research software!

[1] Netherlands eScience Center. (2023). Practical guide to Software Management Plans. Martinez-Ortiz, C., et al. Doi: 10.5281/zenodo.7589725

Reproducible Research Software

What is “Research Software”?

- Software that is itself a **research activity** or **experiment**, a **scholarly object** worthy to be published
- **Code that directly support the research conclusions** in journal publications and pre-prints that arise from agency-supported research. (Tri-Agency Research Data Management Policy, §3.3 Data Deposit)
- Software that supports research, such as **tools** and **infrastructure systems**
- Anything in between

Reproducible Research Software

Reproducibility Goals

- Criteria of different levels of reproducibility suggested by *How reproducible should research software be?*

- **Level 0 - The script you downloaded that one time from the internet**
  - Shared code snippets, functions, single operations from the internet
  - Simple code intended for single or occasional use
  - Needs to be verified for correctness and appropriateness

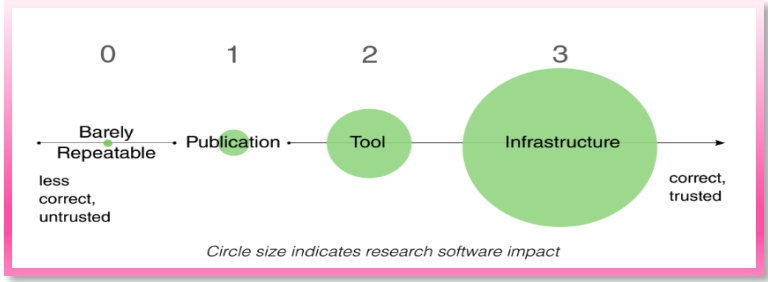


Image retrieved from *How reproducible should research software be?*

Reproducible Research Software

Reproducibility Goals

- Criteria of different levels of reproducibility suggested by *How reproducible should research software be?*

- **Level 1 - Research software for publication**
  - Code that directly support the research conclusions
  - E.g. data processing, analysis, and visualization scripts specific to the project or data
  - Facilitate trust in the published results obtained from research software

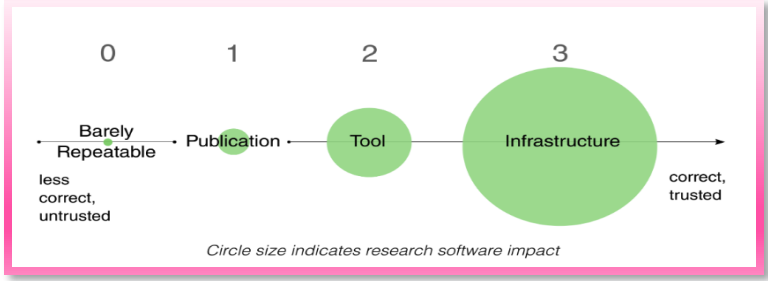


Image retrieved from *How reproducible should research software be?*

Reproducible Research Software

Reproducibility Goals

- Criteria of different levels of reproducibility suggested by *How reproducible should research software be?*

- **Level 2 - Research software as a tool**
  - Code that is intended to be applied to different inputs or scenarios over a modest period of time
  - E.g. Scientific software packages, web development frameworks
  - Either the original developer, or somebody else with the skills to understand and maintain or modify the code themselves

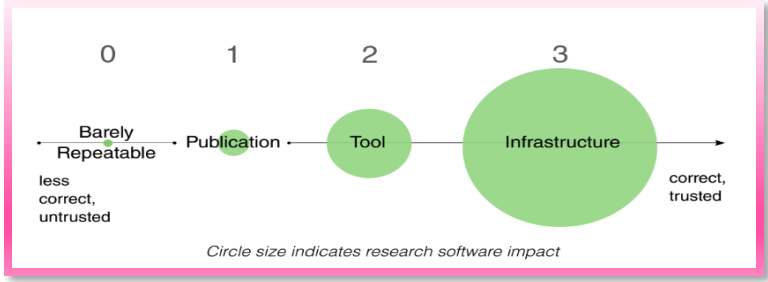


Image retrieved from *How reproducible should research software be?*

Reproducible Research Software

Reproducibility Goals

- Criteria of different levels of reproducibility suggested by *How reproducible should research software be?*

- **Level 3 - Research software as infrastructure**
  - Software that is used by a broader community
  - E.g. web systems, management systems
  - Fundamental aspects include correctness, reusability, and documentation

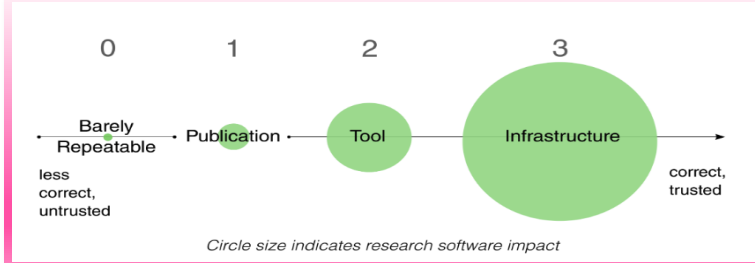


Image retrieved from *How reproducible should research software be?*

Reproducible Research Software

Reproducibility Goals

- Reproducibility goals are proportionate to the size, complexity and audience of the software
- Relates to the expected re-usability of the code and reproducibility of the results
- Level of documentation is proportionate to the reproducibility goals

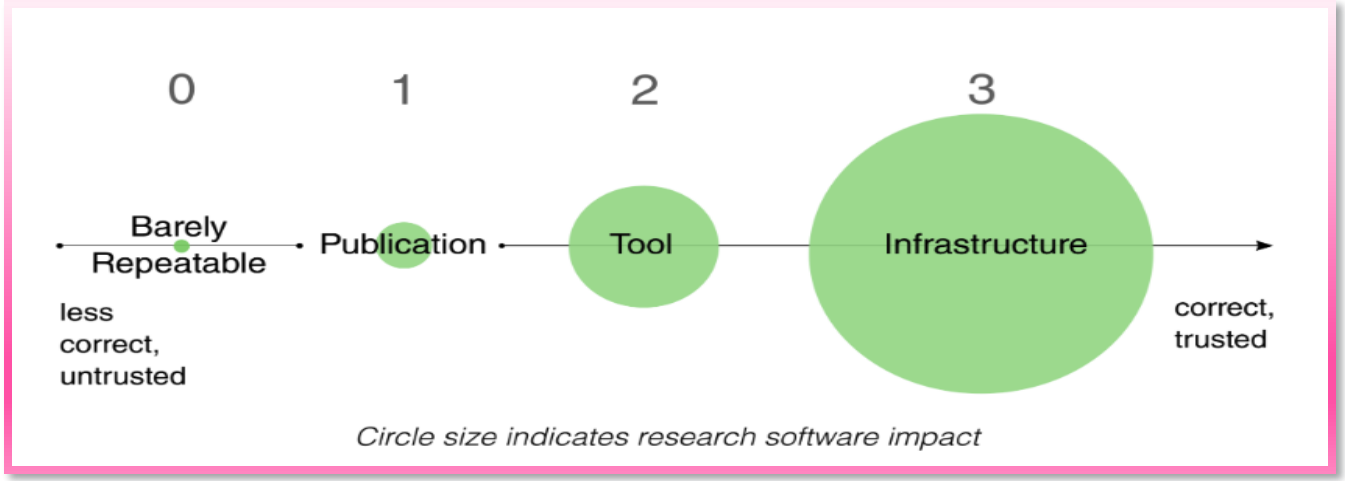


Image retrieved from *How reproducible should research software be?*

Sherman Centre **Certificate Program**

- Certificate you can add to your CV or ORCiD
- Attend 7 RDM workshops to receive a certificate!
- Go to this website to verify today's session: **https://u.mcmaster.ca/verification**
- Learn more about the Certificate Program: **https://scds.ca/certificate-program**

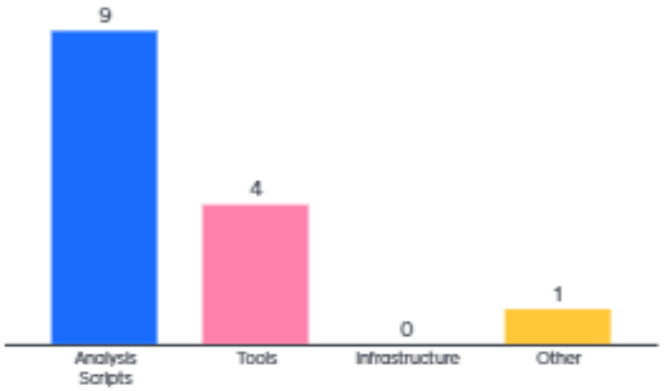
Image by Markus Spiske on Unsplash.

Survey Question

Mentimeter

Join at menti.com | use code 9314 3691

What kind of Research Software do you create?



Reproducible Research Software

Software Management Plan (SMP)

- For **project stakeholders** to reference at anytime
- A **living document,** periodically review during the research process
  - Allow you to consider and define your **reproducibility goal**
  - Provide your research team a **general guideline** to FAIR principle and reproducibility
- Outlines the **strategies** and **practices** for effectively managing software throughout the research process
- Ensures the **continuity of operations** by implementing plans for knowledge retention and knowledge transfer
- Proactively **identifies** and **mitigates** any ethical, licensing and legal risks and liabilities in the project

Reproducible Research Software

Software Management Plan (SMP)

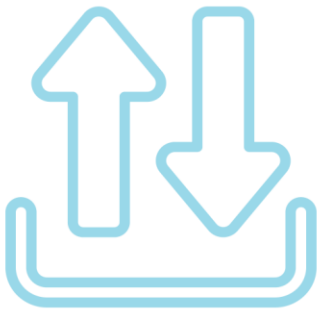
SMP typically includes plans for the following considerations:

- Assets used and produced (input and output)
- Documentation
- Version control protocol
- Preservation and sustainability
- Roles and responsibilities
- Ethics, licensing and legal compliance

Software Management Plan

Assets Used and Produced

- A.K.A. **inputs** and **outputs**
- Input assets used like **research data** and **third-party software libraries**
- These inputs produce **the software source code/executable**, **documentation**, **licenses** and **research outputs**
- Identifies **project components** to project team, stakeholders and future maintainers
- Plans for **compliance with the terms of use** for any third-party data or software libraries integrated into project



Software Management Plan

Documentation

- A software management plan **should** include explanation of how the research software will be documented
- A research software project commonly includes **guides** for both **users** and **developers**
- This documentation helps others understand how to use the software and what it can or should do, as well as how to maintain and extend it

Software Management Plan

Documentation

- **User guide**
  - Assist user in understanding the features
  - Provides instructions, information, and guidance on how to use the software
  - Serves a reference guide during troubleshooting
  - “I’m not getting the expected results. Am I using it correctly, or is there a problem in the code?”

Software Management Plan

Documentation

- **Developer guide**
  - Targeted at other research software developers who may be involved in the development, customization or future maintenance of the software.
  - Provides comprehensive technical documentation
  - Ensures the long-term sustainability of the software project

Software Management Plan

Documentation

```
poly = PolynomialFeatures(degree=2, include_bias=False)
```

- **Documentation** can be simple
  - Just a README file with instructions to build/run the software
  - Example: OpenAI/Whisper, MakeAReadme.com
  - Or exist as comments in the source code
- **Code commenting** is the practice of adding short notes throughout your code
- The goal is to publish code that can be **understood** by others
- Publish **well-documented**, **well-commented** code
- There are **tools** to assist with code commenting, e.g. **GitHub Copilot**

speech translation, spoken language identification, and voice activity detection. These tasks are jointly represented as a sequence of tokens to be predicted by the decoder, allowing a single model to replace many stages of a traditional speech-processing pipeline. The multitask training format uses a set of special tokens that serve as task specifiers or classification targets.

Setup

We used Python 3.9.9 and PyTorch 1.10.1 to train and test our models, but the codebase is expected to be compatible with Python 3.8-3.11 and recent PyTorch versions. The codebase also depends on a few Python packages, most notably OpenAI's tiktoken for their fast tokenizer implementation. You can download and install (or update to) the latest release of Whisper with the following command:

```
pip install -U openai-whisper
```

Alternatively, the following command will pull and install the latest commit from this repository, along with its Python dependencies:

```
pip install git+https://github.com/openai/whisper.git
```

To update the package to the latest version of this repository, please run:

```
pip install --upgrade --no-deps --force-reinstall git+https://github.com/openai/whisper.git
```

It also requires the command-line tool `ffmpeg` to be installed on your system, which is available from most package managers:

```
# on Ubuntu or Debian
sudo apt update && sudo apt install ffmpeg

# on Arch Linux
sudo pacman -S ffmpeg

# on MacOS using Homebrew (https://brew.sh/)
brew install ffmpeg

# on Windows using Chocolatey (https://chocolatey.org/)
choco install ffmpeg

# on Windows using Scoop (https://scoop.sh/)
scoop install ffmpeg
```

You may need `rust` installed as well, in case tiktoken does not provide a pre-built wheel for your platform. If you see installation errors during the `pip install` command above, please follow the Getting started page to install Rust development environment. Additionally, you may need to configure the `PATH` environment variable, e.g. `export PATH="$HOME/.cargo/bin:$PATH"`. If the installation fails with `No module named 'setuptools_rust'`, you need to install `setuptools_rust`, e.g. by running:

Software Management Plan

Version Control Protocol

- Version control protocol provides a **systematic approach** to manage development and collaboration
- **Audit trail** of changes with reasons
- Version control helps keep your code base **transparent** and **accessible**.
- Exist **cloud services** to make all of this easier, e.g. **GitHub**, which can provide:
  - Automated change/difference identification
  - Backups of each change and version
  - Shared public and private remote access for collaborators
  - Conflict resolution when editing the same file

Software Management Plan

Version Control Protocol

master linux / fs / sync.c

Code Blame 391 lines (344 loc) · 10.4 KB Code 55% faster with GitHub Copilot

Older Newer

7 years ago	License cleanup: add SPDX GPL-...	1	`// SPDX-License-Identifier: GPL-2.0`
19 years ago	[PATCH] sys_sync_file_range()	2	`/*`
		3	` * High-level sync()-related operations`
		4	` */`
		5	
3 years ago	block: remove __sync_blockdev	6	`#include <linux/blkdev.h>`
19 years ago	[PATCH] sys_sync_file_range()	7	`#include <linux/kernel.h>`
		8	`#include <linux/file.h>`
		9	`#include <linux/fs.h>`
14 years ago	include cleanup: Update gfp.h a...	10	`#include <linux/slab.h>`
13 years ago	fs: reduce the use of module.h w...	11	`#include <linux/export.h>`
13 years ago	introduce sys_syncfs to sync a si...	12	`#include <linux/namei.h>`
18 years ago	[PATCH] severing fs.h, radix-tree....	13	`#include <linux/sched.h>`
19 years ago	[PATCH] sys_sync_file_range()	14	`#include <linux/writeback.h>`
		15	`#include <linux/syscalls.h>`
		16	`#include <linux/linkage.h>`
		17	`#include <linux/pagemap.h>`
18 years ago	[PATCH] BLOCK: Move functions ...	18	`#include <linux/quotaops.h>`
14 years ago	Catch filesystems lacking s_bdi	19	`#include <linux/backing-dev.h>`

Software Management Plan

Preservation & Sustainability

- Involves measures to address technological, organizational and environmental factors
  - **Technological**: Advances in hardware technology or changes in host systems can impact software compatibility
  - **Organizational**: Project stakeholders (e.g. staff, developers) changes
  - **Environmental**: Changes in regulatory compliance, e.g. AODA
- Long-term Availability, Usability, and Maintenance of software beyond project's duration
- Continuity of Operations
  - Knowledge Retention
  - Knowledge Transfer
  - Preservation allows future maintainers to understand the software's history and decision-making processes

Software Management Plan: Preservation & Sustainability

Preserving Software and Dependencies

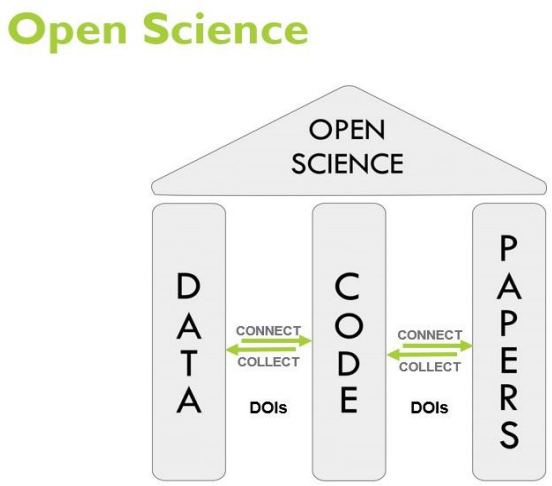
- Store your source code with dependencies
  - Package manager to install, manage and document (e.g. `pip`, `conda`, `npm`)
- Building and sharing software as container images
  - Via Docker Hub or self-hosted container image repository

```
> npm list
app@0.1.1 /home/kelvin/MacScribe/app
├── @azure/msal-browser@2.37.0 extraneous
├── @azure/msal-common@13.0.0 extraneous
├── @azure/msal-react@1.5.7 extraneous
├── @emotion/react@11.10.6
├── @emotion/styled@11.10.6
├── @mui/icons-material@5.11.11
├── @mui/material@5.11.14
├── @mui/x-data-grid@6.0.4
├── @next/font@13.1.6
├── @types/node@18.13.0
├── @types/react-dom@18.0.10
├── @types/react@18.0.27
├── eslint-config-next@13.1.6
├── eslint@8.33.0
├── gray-matter@4.0.3
├── mui-file-input@2.0.0
├── next@13.1.6
├── prettier@2.8.8
├── raw-loader@4.0.2
├── react-dom@18.2.0
├── react-markdown@8.0.6
├── react@18.2.0
├── styled-components@5.3.9
└── typescript@4.9.5
```

```
> pip freeze
attrs==21.2.0
Automat==20.2.0
bcrypt==3.2.0
blinker==1.4
certifi==2020.6.20
chardet==4.0.0
click==8.0.3
colorama==0.4.4
```

Software Management Plan: Preservation & Sustainability

Preserving Software with Persistent Identifiers

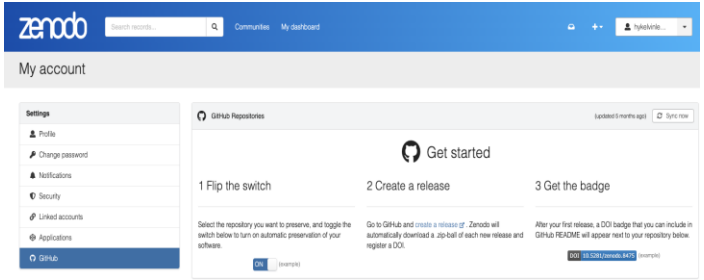


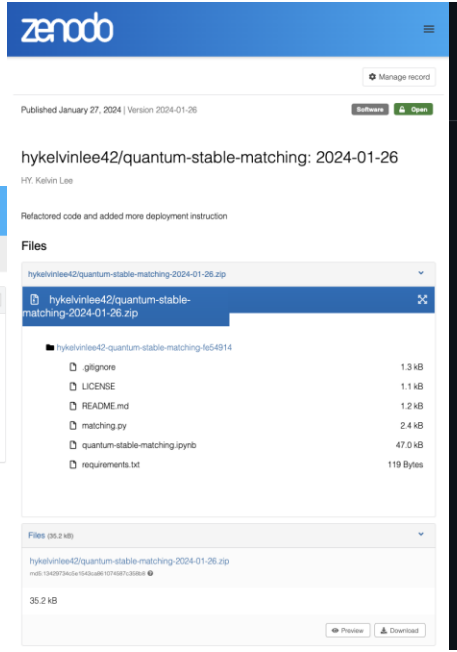
Courtesy of Jeffrey Demaine

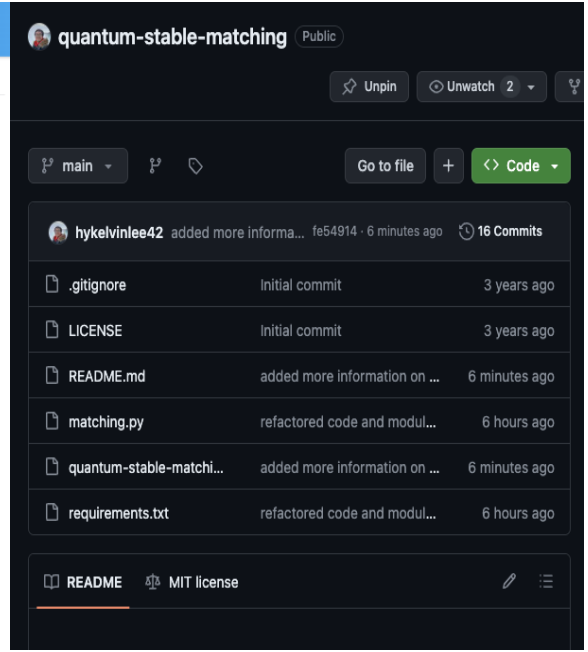
- Generating (persistent) Digital Object Identifiers for your software and source code
  - DOIs allow software to be easily referenced in publication
  - DOIs will improve findability of the software in the research community
- Using Zenodo and GitHub
  - Zenodo example: QSM
  - GitHub example: QSM

Software Management Plan: Preservation & Sustainability

Preserving Software with Persistent Identifiers







Software Management Plan: Preservation & Sustainability

Software Testing

- Testing aids in managing dependencies by verifying the software works correctly and provides a shared understanding of the software's behaviour
- A test plan should be well-documented, well-commented
- Formal test plan includes:
  - Unit Testing (test a function works as intended)
  - Regression Testing (re-run old tests to make sure nothing else changed)
  - Integration Testing (test system, i.e. all of the functionality together)
- Using Continuous Integration (CI) Automation Tools
  - GitHub Actions
  - Travis CI

Software Management Plan

Ethics, Licensing & Legal Compliance

- Consider potential ethical biases in software (algorithms) developed/used to process your data
  - Does your software handle data in a way that respects privacy rights and maintains confidentiality
  - Does your software provide results from an unbiased perspective
- Consider software licensing as early as possible
  - Consider how others may use your software
  - Your software dependencies (assets used) may have licenses restricting how or others can share or reuse your software, which can have license requirements for your work
  - **If you don't include a software license, nobody else can copy, distribute, or modify your work**

Software Management Plan

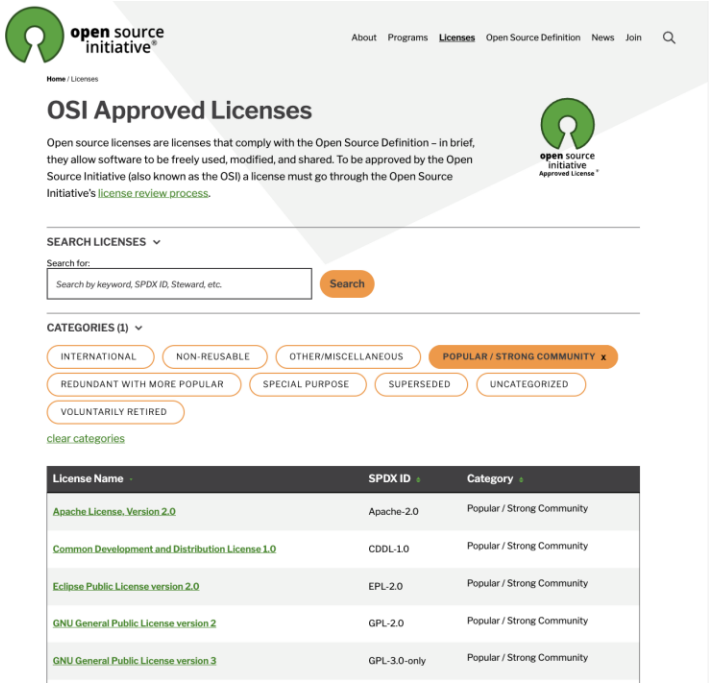
Ethics, Licensing & Legal Compliance

- Where possible, use existing licenses, rather than creating custom terms of use
  - Common OSS Licenses like GNU GPL, MIT, Mozilla and Apache
  - Existing licenses are designed to be (mostly) compatible with other commonly used licenses
  - Creating custom licenses requires careful consideration of legal and licensing issues
- Web tools to help:
  - Choose-a-license (choosealicense.com)
  - Open Source Initiative (opensource.org/licenses)

open source initiative
Approved License ®

Software Management Plan

Ethics, Licensing & Legal Compliance



open source initiative

About Programs Licenses Open Source Definition News Join

Home / Licenses

OSI Approved Licenses

Open source licenses are licenses that comply with the Open Source Definition – in brief, they allow software to be freely used, modified, and shared. To be approved by the Open Source Initiative (also known as the OSI) a license must go through the Open Source Initiative's license review process.

open source initiative Approved License

SEARCH LICENSES

Search for:

Search by keyword, SPDX ID, Steward, etc.

Search

CATEGORIES (1)

INTERNATIONAL | NON-REUSABLE | OTHER/MISCELLANEOUS | POPULAR / STRONG COMMUNITY x | REDUNDANT WITH MORE POPULAR | SPECIAL PURPOSE | SUPERSEDED | UNCATEGORIZED | VOLUNTARILY RETIRED

clear categories

License Name	SPDX ID	Category
Apache License, Version 2.0	Apache-2.0	Popular / Strong Community
Common Development and Distribution License 1.0	CDDL-1.0	Popular / Strong Community
Eclipse Public License version 2.0	EPL-2.0	Popular / Strong Community
GNU General Public License version 2	GPL-2.0	Popular / Strong Community
GNU General Public License version 3	GPL-3.0-only	Popular / Strong Community

Home

Appendix

For reference, here is a table of every license described in the choosealicense.com repository.

If you're here to choose a license, start from the home page to see a few licenses that will work for most cases.

License	Commercial use	Distribution	Modification	Patent use	Private use	Disclose source	License and copyright notice	Network use is distribution	Same license	State changes	Liability	Trademark use	Warranty
BSD Zero Clause License	●	●	●		●						●		●
Academic Free License v3.0	●	●	●	●	●		●			●	●	●	●
GNU Affero General Public License v3.0	●	●	●	●	●	●	●	●	●	●	●		●
Apache License 2.0	●	●	●	●	●		●			●	●	●	●
Artistic License 2.0	●	●	●	●	●		●			●	●	●	●
BSD 2-Clause "Simplified" License	●	●	●		●		●				●		●
BSD 3-Clause Clear License	●	●	●	●	●		●				●		●
BSD 3-Clause "New" or "Revised" License	●	●	●		●		●				●		●
BSD 4-Clause "Original" or "Old" License	●	●	●		●		●				●		●
Boost Software License 1.0	●	●	●		●		●				●		●
Creative Commons Attribution 4.0 International	●	●	●	●	●		●			●	●	●	●
Creative Commons Attribution Share Alike 4.0 International	●	●	●	●	●		●		●	●	●	●	●
Creative Commons Zero v1.0 Universal	●	●	●	●	●						●	●	●
CeCILL Free Software License Agreement v2.1	●	●	●	●	●	●	●	●	●		●		●
CERN Open Hardware Licence Version 2 - Permissive	●	●	●	●	●		●			●	●		●
CERN Open Hardware Licence Version 2 - Strongly Reciprocal	●	●	●	●	●	●	●		●	●	●		●
CERN Open Hardware Licence Version 2 - Weakly Reciprocal	●	●	●	●	●	●	●		●	●	●		●
Educational Community License v2.0	●	●	●	●	●		●			●	●	●	●
Eclipse Public License 1.0	●	●	●	●	●	●	●		●		●		●
Eclipse Public License 2.0	●	●	●	●	●	●	●		●		●		●
European Union Public License 1.1	●	●	●	●	●	●	●	●	●	●	●	●	●
European Union Public License 1.2	●	●	●	●	●	●	●	●	●	●	●	●	●
GNU Free Documentation License v1.3	●	●	●		●	●	●		●	●	●		●
GNU General Public License v2.0	●	●	●		●	●	●		●	●	●		●

Reproducible Research Software

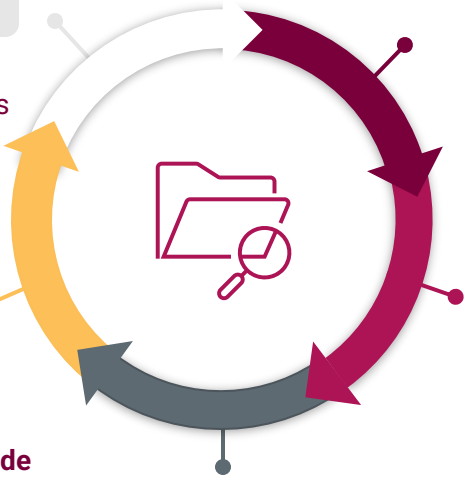
Software Management Plan (SMP)

Elements of an SMP:

- Assets used and produced (input and output) ✓
- Documentation ✓
- Version control protocol ✓
- Preservation and sustainability ✓
- Roles and responsibilities (same as DMP)
- Ethics, licensing and legal compliance ✓

Reproducible Research

Software and Data Management Plans are an integral part of good research data management practices. They lay out your plan to create, store, organize, document, secure, preserve, and share your research software and data. It's an essential part of the research data lifecycle and good data management practices.



Preservation

- FAIR principles
- Persistent identifiers
- **Data repositories**
- **Software source code repositories**
- Open file formats
- Metadata

Publication

- **Data sharing**
- **Software source code sharing**

Planning

- **Data Management Plans**
- **Software Management Plans**
- Grant applications
- Ethics applications
- Data sharing agreements
- Data access

Data Collection

- Electronic Lab Notebooks
- De-identification
- Data documentation
- File management
- Data storage
- Data security

Analysis

- **Analysis scripts**
- **Analysis documentation**

Software Management Plan Templates

Samples and Templates

- Our McMaster Software Management Plan template on DMP Assistant
- DMP Assistant: https://dmp-pgd.ca/
- A Data Management Plan and a Software Management Plan can be used together to plan the management of these research assets

Software Management Plan: Testing DMP Assistant

Project Details | Contributors | Plan overview | Write Plan | Research Outputs | Share | Request feedback | Download

McMaster Software Management Plan

This plan is based on the "McMaster Software Management Plan" template provided by McMaster University.

This template is provided by the McMaster University Research Software Development Team for general SMP creation. This template is based on the McMaster General Purpose DMP Template and is not designed for any particular discipline.

Template version 0, published on January 30, 2024

Instructions

Write plan

Documentation and Metadata
- What documentation (user manual, develop manual, features documentation e.g.) will be needed for the software to be built, operated and modified correctly in the future?
- What tools and standard (README.md, requirements.txt, package.json, Dockerfile e.g.) are you using to document and describe your software, its dependancies and its build environment?

Storage and Backup
- What are the anticipated storage requirements for your project, in terms of storage space (in megabytes, gigabytes, terabytes, etc.) and the length of time you will be storing it?
- How and where will your data and software be stored, backed up and version-controlled during your research project?
- How will the research team and other collaborators access, modify, and contribute data and software throughout the project?

Preservation
- How will you manage your software for long-term preservation and access at the end of your research project? Will you deposit your software in a source code repository management service, container images repository (Docker Hub e.g.) or a software package registry (pip, npm e.g.)?
- Indicate how you will ensure your software is preservation-ready. Consider preservation-friendly file formats, ensuring matching dependencies, and inclusion of supporting documentation.
- Are there testing or other validation procedures included as documentation or part of the source code?
- How will you manage internal and external issue tracking/reporting?

Sharing and Reuse
- What software components will you be sharing and in what form? (source code, APIs e.g.).
- Have you considered what type of end-user agreement and sharing license to include with your software? If so, Which end-user agreement and sharing license will you include with your software.

Responsibilities and Resources
- Identify who will be responsible for managing this project's data and software during and after the project and the major data and software management tasks for which they will be responsible.
- How will responsibilities for managing data activities be handled if substantive changes happen in the personnel overseeing the project's data, including a change of Principal Investigator?

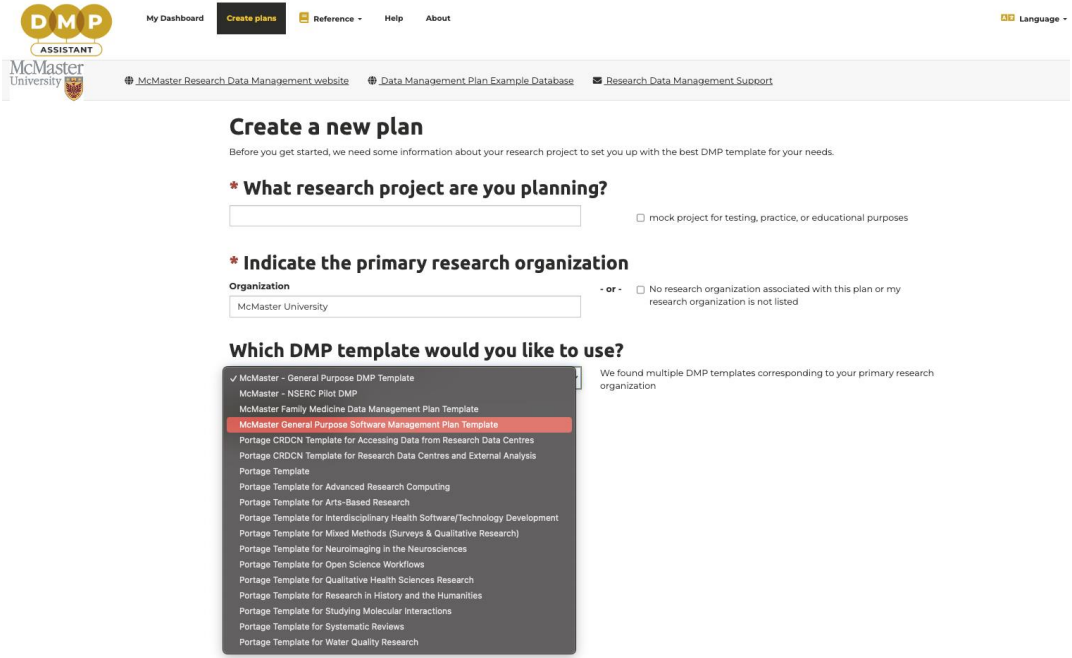
Ethics and Legal Compliance
- Are there any potential biases presented in the algorithms/models of the software you developed?
- Will your research software be handling any sensitive data? What security measures did you implement into the software to adhere to data protection regulations?
- How will you manage legal, ethical, and intellectual property issues?

DMP ASSISTANT PGD

Software Management Plan Templates

McMaster General Purpose Software Management Template

- Create a new management plan on DMP Assistance
- Input "McMaster University" as the primary research organization
- You will then find "McMaster General Purpose Software Management Plan Template" in the template dropdown menu



Best Practices for Managing Your Code and Scripts to Generate Your Research

Please reach out to us if you have questions about the elements of Software Management Plans (RSD team) or Data Management Plans (RDM team).

- Research Software Development (RSD) Team
  - Email: rsd@mcmaster.ca
  - Website: https://u.mcmaster.ca/rsd
  - Join our community of practice: https://u.mcmaster.ca/rsd-ms-team
  - Reproducible Research Software Learning Module
- Research Data Management (RDM) Team
  - Email: rdm@mcmaster.ca
  - Website: htts://rdm.mcmaster.ca
  - Join our community of practice: https://u.mcmaster.ca/rdm-community

Thank You!

McMaster University Research Software Development and Research Data Management Teams