deRSE19 - FAIR Software BoF

Potsdam, 5/6/2019

This document is also available via https://tinyurl.com/deRSE-FAIRSoftware

Introductory slides to the session:
https://drive.google.com/file/d/1oGEIW5bxh5KdM4yWPvbPiGeQ13hG_khT/view?usp=sharing

Participants

We started the session with a quick round of introduction: *Who are you and why are you here?*

- Anna-Lena Lamprecht (Utrecht University) -- software engineering researcher and educator caring about better research software
- Carlos Martinez (Netherlands eScience Center) -- RSE. Mission: improving the quality of research software.
- George (Physics) -- interoperable? Share principles with Julia community?
- Person X (Humanities) -- digital practices
- Christian -- fit FAIR / FOSS
- Svantje Lilienthal (TIB) -- FAIR data sources https://labs.tib.eu/rosi/tech.php GitHub
- Katrin Leinweber (TIB.eu) -- bring FLOSS habits into RSE #GitLab4Science
- Hendrik Geßner (University of Potsdam, CRC 1294) - sustainable software and FAIR
- Stefan - open data and software, FAIR
- Katja -- general interest
- Marc -- done FAIR stuff in the past
- Michele -- Documentation
- Andreas -- find out more about FAIR, and what does it mean for RS
- Henning -- sustainable research
- Neil Chue Hong (Software Sustainability Institute) -- advocate for foundational principles of FAIR for software, provide guidance for researchers, understand what people think I and R mean for software

Discussion

We then split into three groups discussing different aspects of FAIR research software (FAIR and FOSS, FAIR and Sustainability, practical FAIR things). The main findings of these groups are summarized below.

FOSS (Free and Open Source Software) and FAIR

- FAIR and FOSS mainly overlap with regard to their intentions about reusability.
- Open access is probably not included in FAIR (data) principles due to privacy concerns. However, these concerns are not valid for software.
- In general, there is little knowledge about licensing and the impacts on further usage in the scientific community.
- The concept of dependencies between licences for different objects is already present in FAIR, as principle I2 formulates that vocabularies used by data objects must also comply to the FAIR criteria, so that interchangeability is guaranteed.

FAIR and Sustainability

- By sustainable software we mean software that is longer usable, maintainable and inspectable. Archival (as in source code repositories or the Software Heritage project) is a means for keeping software inspectable, but to be sustainable for research purposes it has to be more than that, namely usable and maintainable.
- The checklist for FAIR software on slide 8 of the Neil's presentation "FAIR enough? Can we (already) benefit from applying the FAIR data principles to software?" includes some good/best practices for making software FAIRer
- We think it is important to distinguish between the sustainability of the code/implementation vs. the functionality that it provides.
- One question is whether the Interoperability of FAIR for software means that any particular implementation of software could be replaced by another - sustainability of the function of the software. If software is FAIR, it is replaceable and interchangeable.
- Interoperable is more than just creating standards for data but also documenting (in a machine readable way) of how you use them. Interoperable for software would mean that APIs are well documented, not only for humans but for machines, so that other software / tools could understand how to use, replace and interchange them.
- Thinking about existing examples: Docker has a fairly well defined input, a port it exposed to the outside world, but we don't need to understand what goes on inside. In the past, a similar thing was done with Web Services (but you needed someone to host them). However, what happens in 5, 10, 25 years? Docker might change the way that they present input ports.
- Another question is how maintainability relates to FAIR. For data, maintainability can be more easily expressed by just using standard, well-described formats and put in place checksums to identify bitrot. You may also have to document the collection methods.
- For software, the case is not so clear. We might say that the data format corresponds to the algorithm, and the collection methods to software design decisions, and that they should be documented accordingly. This is an interesting point that needs further discussion.

- It might also be a viable approach to create a list of things that need to be documented to keep software sustainable in a FAIR context, such as
  - APIs
  - Data formats
  - Algorithm
  - Dependencies
  - Programming languages
  - *Others?*

Practical things

Some previous works on practical FAIR things can be found here:
- TOP 10 FAIR Things: https://zenodo.org/record/2555498
- Checklist for FAIR software: https://figshare.com/articles/FAIR_enough_Can_we_already_benefit_from_applying_the_FAIR_data_principles_to_software_/7449239 - see checklist on slide 8
- more workshop material about "FAIR Data & Software": https://events.tib.eu/fair-data-software/2018/
- TIBs interpretation of FAIR Software: https://av.tib.eu/series/530/

FAIR is trendy and nice for funders, but as RSEs we should maybe be more concerned about software quality. So, regardless of what's in FAIR, we as RSEs should care about:
- Tests
- Code quality
- Documentation

There is already quite some work on how to make code quality measurable, and what type of metrics are required therefore, such as:
- GitHub Insights: https://github.com/github/opensource.guide/community
- ReadMe Analyzer: https://demos.algorithmia.com/github-readme-analyzer/
- https://help.github.com/en/articles/configuring-automated-security-fixes
- https://sonarcloud.io/about
- Automated best-practices checks in R: https://github.com/MangoTheCat/goodpractice
- Automation: https://travis-ci.org/ or GitLab CI

License advice can be found, e.g., at:
- https://tldrlegal.com/
- http://choosealicense.com/

Finally, two practical questions that remain to be discussed:
- Risk management -- is vendor lock a risk? Is dependency on open source a risk?
- Interoperability -- is it mostly about protocols?