Navigation tree Vr. DRAFT-1

Version: DRAFT-1
Status: Closed for discussion
Based on discussion on the first 3 versions of the Good practices for research software
Following discussion available
Version history available
Contributors: Full list of contributors (please add your name there if you contribute to this navigation tree)

Guidelines navigation tree

It was decided during the April 30th WG call that the goal here is a navigation tree that will make it easier for readers to find the essence of the recommendations document and find the sections relevant for them.

Suggestion for navigation tree

Text-based tree (for brainstorming purposes)

Notes

- Please do not use lists for examples, rather keep them together. A list might give the idea that all possibilities are listed, which is not true for examples
- We want to have at least three sections per recommendation: how to, why is this important, getting more information
- As much as possible, the actual content should be the same as the one provided in the recommendations document
- Please be aware that the actual tree will have the same content but its nodes might differ as a visual representations allows for more creativity

Suggested color code:

- Section
- Recommendation
- Specific “Do”
- Why is this important?
- Comment/explaining notes
- Further information

Based on 2020.05.05 Software group meeting and 7th May report

- Good practices for research software
    - Who are these recommendations for?

        - These recommendations will be relevant to audiences ranging from researchers and research software engineers with comparatively high levels of knowledge about software development to experimentalists, such as wet-lab researchers, with almost no background in software development.
    - ~~What is research software?~~
        - ~~Software is very broadly defined here~~
        - ~~Any piece of code that performs calculations or helps analyse data is understood to be software in this recommendation~~
        - ~~Examples:~~
            - ~~Algorithms developed for exploration of data, e.g., R scripts~~
            - ~~Libraries used in other software (e.g. Python functions or R libraries)~~
            - ~~Macros (e.g., in an Excel spreadsheet)~~
            - ~~Scripts that run on analysis platforms (e.g., Matlab, R, Python, Julia, bash, etc)~~
            - ~~Standalone applications with a single binary, e.g. C++ code~~
            - ~~Web-based applications written in e.g. Javascript~~
            - ~~Workflows (i.e., the sequence of connected tasks in the order defined by flow and/or data dependencies). (e.g. Galaxy, Taverna, WCL, Pegasus)~~
        - ~~Four recommendations to encourage best practices in research software~~
    - Setup: get things organised for you and others
        - R1: Make your software available
            - How to
                - Make it available via code repositories
                - Prefer those with version control
                - There are many alternatives, e.g., GitHub, GitLab, Bitbucket
                - Getting more information
                    - 4OSS
                    - Guidelines on code repositories
            - Why is this important?
                - Version control makes tracking changes easier, and you can easily go back when needed
                - You can get all your code in one place
                - Others can find your code, reuse it and contribute to it
                - Sharing your code enables reuse, it helps advance science
        - R4: Ensure reproducibility and portability of your results
            - How to
                - Document all parameter values (including setting random seeds to predetermined values)

        - Provide all information required to set up the right environment to run your code
        - You can go one step further and use containers such as Docker or Singularity
        - Getting more information
            - Ten Simple Rules for Writing Dockerfiles for Reproducible Data Science
    - Why is this important?
        - Making it easier for others to reproduce your code increase trust and credibility
        - Packaging your software in a container such as Docker or Singularity helps others install the software and ensure that they run it in the same environment as you do
- Reuse: build upon and help others to do so
    - R3: Provide metadata/documentation for others to use your software
        - How to
            - ~~For complex systems, include a high-level system architecture diagram~~
            - Add comments explaining your thought process in your code
            - List libraries/tools that you use and their version numbers
            - Include sample input/output files. If they are also available in public datasets, you can add a link to it but saving a local copy is always a good idea
            - Getting more information
                - Ten simple rules for documenting scientific software
        - Why is this important?
            - If exact parameters or the details of the compute environment are not known, it will be impossible for others to reproduce your results
            - The better others can understand how your software works, the easier it becomes for them to build upon it
    - R5: Release your software under a license
        - How to
            - Code repositories mentioned on R1 will make it easier for you to select a licence, a “LICENSE” file will be automatically added to your repository
            - If possible, use an open source-license
                - How to choose an open source license
            - Be aware of the difference between copyleft and copyright licenses
                - Copyleft licenses include a clause that guarantees continued open access to a software and its source

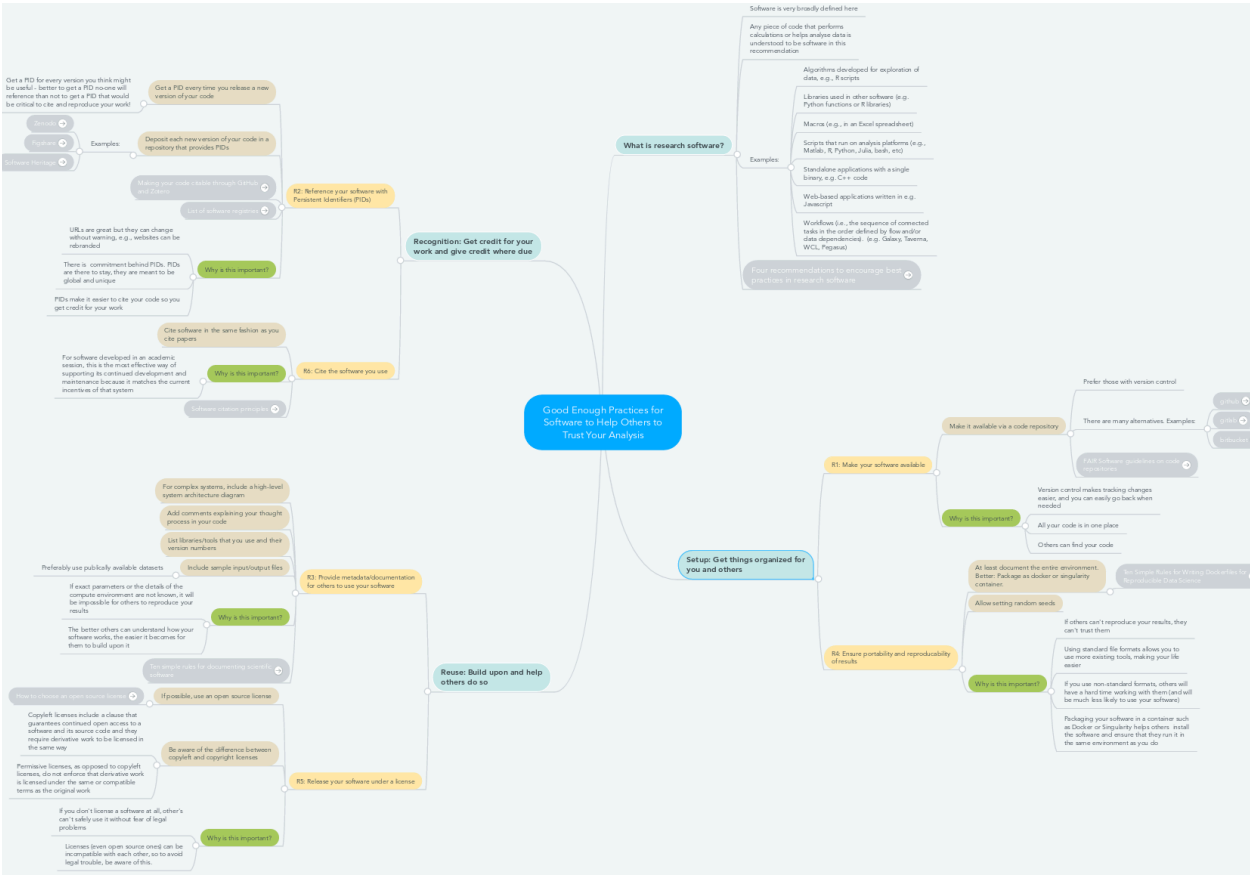
code and they require derivative work to be licensed in the same way
                - Permissive licenses, as opposed to copyleft licenses, do not enforce that derivative work is licensed under the same or compatible terms as the original work
        - Why is this important?
            - If you don't license a software at all, others can't safely use it without fear of legal problems
            - Licences (even open source ones) can be incompatible with each other, so to avoid legal trouble, be aware of this.
- Recognition: Get credit for your work and give credit where due
    - R2: Reference your software with Persistent Identifiers (PIDs)
        - How to
            - Deposit each new released version of your code in a repository that provides PIDs
            - Examples: Zenodo, Figshare, Software Heritage
            - Getting more information
                - FAIR software guidelines on citing software
                - List of software registries
                - Making your code citable through GitHub and Zenodo
        - Why is this important?
            - Repositories providing PIDs provide persistent storage
            - PIDs make it easier to cite your code so you get credit for your work
            - Better to get a PID even if you are not sure someone will reference than not to get a PID that would be critical to cite and reproduce your work!
    - R6: Cite the software you use
        - How to
            - Cite software in the same fashion as you cite papers
            - Remember to include the version you actually used
            - If there is a PID available for the software you are using, use it!
            - Getting more information
                - Software citation principles
        - Why is this important?
            - For software developed in an academic session, this is the most effective way of supporting its continued development and maintenance because it matches the current incentives of that system

Using Mindmeister

Initially, I suggest building that as a mind map, since that allows the easiest editing (especially drag&dropping nodes around and collapsing subtrees to hide completed subtrees). Then we can think about what to export that as or if we want to transfer it to a different visualization.
Links:

- Editing: https://mm.tt/1491557723?t=BMblC84jk7
- Viewing: https://www.mindmeister.com/1491557723/working-with-omics-data

Image of what it looks like now:



Legend:

- Blue background: root node
- White background: Navigation through the recommendations
- Yellow background: Specific recommendations
- No background: Comments/explanations for the recommendations
- Grey background: Links to external supporting resources (click on the arrow in the box to open the link)

I have added the first part of the software recommendations. If that seems like a good way to move forward, we can do the whole document that way.

(historical) Guidelines decision tree

This part of the document is kept for historical reasons/reference, the currently active discussion points are on the pages above.

This document corresponds to the task:

- Creating a decision tree to help the research community to apply the guidelines. This is one of the 3 stated outputs that RDA wish to achieve, although I think only the clinical subgroup has done this so far (see attached slide 17 of 21 April RDA webinar)

Deadlines

- Discussion: What sort of tree do we want to provide?: 5th of May
- Draft ready for final discussion: 8th May
- Comments on draft trees: 12th May
- Official final deadline: 15th May

Call for discussion

There is no clarity yet on who and what a decision tree regarding the “good practices for research software” guidelines should support. Please help us clarify the scope (could be of course multi-scope). Here is what we have so far

- A set of navigation tree summarizing the guidelines in a graphical way (same stakeholders as for the guidelines, e.g, not necessarily research software developer)
    - One branch regarding guideline compliance
    - One branch regarding resources supporting a guideline
    - One branch regarding resources providing more information on the topic
    - Example targeting Guideline R1
- A set of decision trees helping research software developers to apply the guidelines based on tasks that should be carried out at the ideally at beginning of a new development
    - Question 1: How to choose a license? Guidelines R5 and R6
    - Question 2: How to choose a code repository? Guideline R1
    - Question 3: How to get a PID? Guideline R2
    - Question 4: How to describe my software? → metadata (including provenance and references to others) and documentation. Guidelines R3 and R7
    - Question 5: How to increase reproducibility? Guideline R4
    - Example targeting question 1
- Any other options?

- Discussion points from the main document

    - Thread 1
        - LJ Garcia at ZBMED 4:26 PM Apr 24. Piotr, thanks! I started a draft, please have a look to the link, just trying to get the key points from the guidelines +barkermd@outlook.com, +fpsom@certh.gr, +Hugh.Shanahan@cs.rhul.ac.uk. I am finding it difficult to come up with a tree that will end with a global "yes software complies to guidelines" or "no, it does not", not sure if that is the idea. Is it ok if we have branches per Recommendation? At least for those that are independant, e.g., I can have a PID but do not provide the source code (so no code repository)
        - Piotr Wojciech Dabrowski 9:34 PM Apr 26. Thanks, happy to work on that with you :D Maybe we could have the goal as a question during the next RDA-COVID19-omics call (will you be there? I'm torn between several appointments, but if you can't make it, I will do my best)? I am not entirely clear on who will be the target audience of the tree (Reviewer? Developer? Curator? PI? Policy maker?) and what the tree is supposed to achieve (test compliance? Show next development steps? Help work through the process of sharing the software?). I think that needs to be clearly defined before we can actually build a helpful tree.

Notes from WG call on April 30th, 14:00-15:00

Question points to be asked:

- Who do we want to be using the decision tree?
- What is to be decided using that decision tree?
- Ideally: What other decisions are to be supported based on that?

Current ideas:

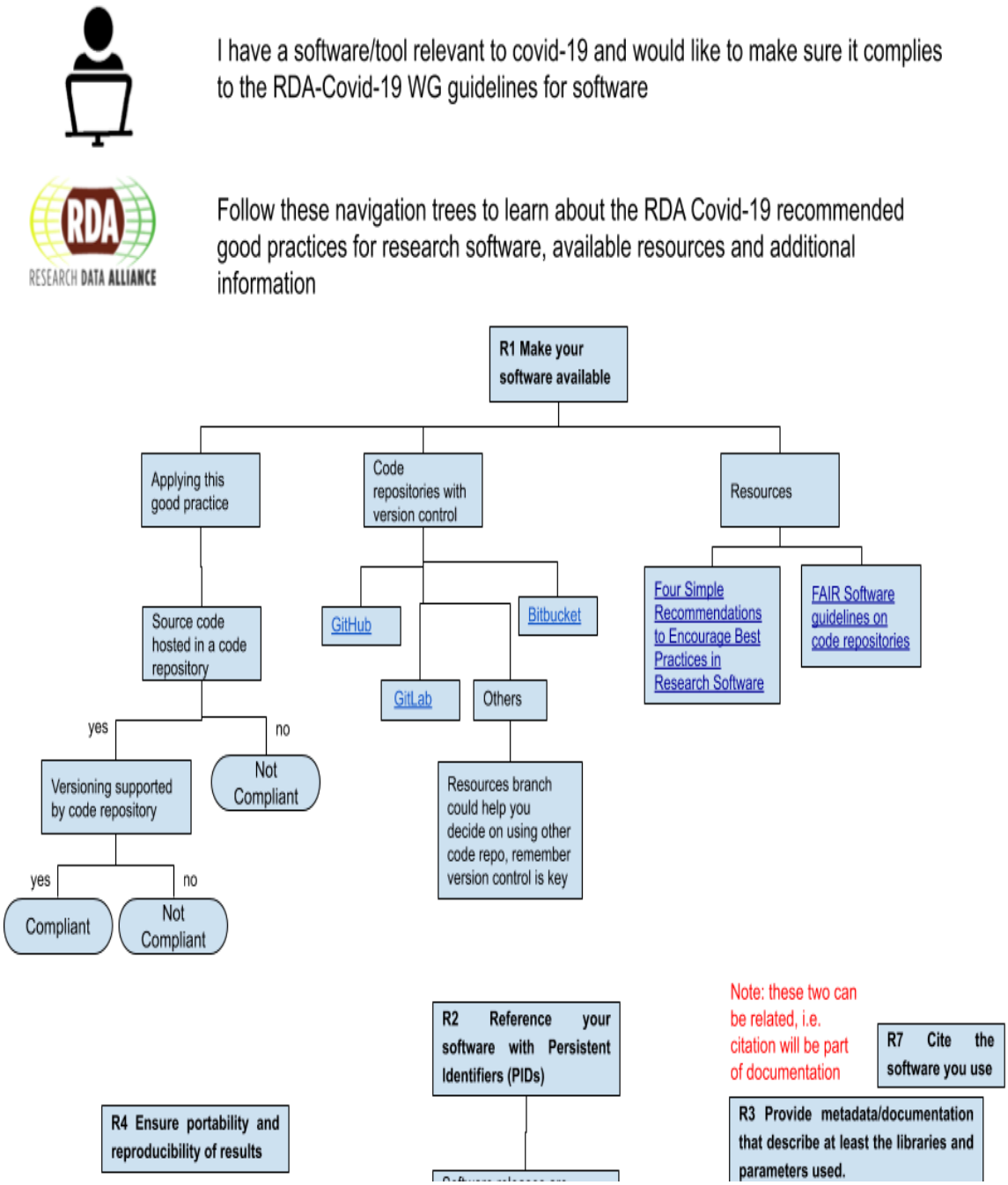
- Navigation tree (hierarchical checklist) for guiding through the recommendations, with three branches:
    - Guideline compliance
    - Other resources providing related/further guidance
    - Other resources providing more general information on the topic
- Decision tree helping research software developers apply some of the guidelines (those where a decision tree can help), similar to but more comprehensive than that being published by RSE:
    - Level 1: How to choose a license (Recommendation R5 and R6)
    - Level 2: How to choose a core repository (Recommendation R1)
    - Level 3: How to get a PID (Recommendation R2)

Feedback from the group: Do a navigation tree to help readers through the complex document and then branches out into a decision tree whenever that is necessary/helpful.

Ideas

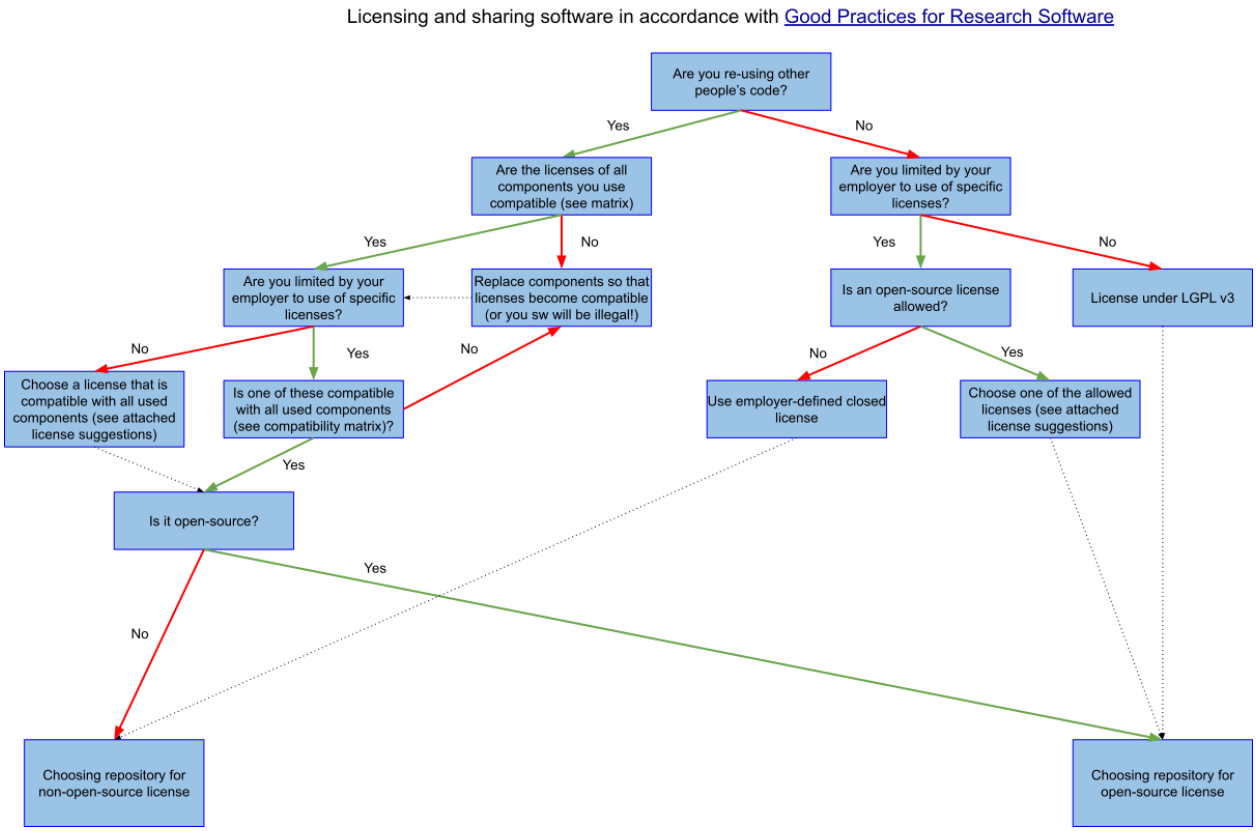
Navigation tree summarizing guidelines

Drawing located at
https://docs.google.com/drawings/d/1dpmiAx0vrJkXNaZDlNMq0CfBrfur0rjDYkK7zviAa7U/edit



Deprecated: Decision tree to choose elements mentioned on the guidelines

This has been decided against as a main tree, but keeping it in here since it might still be useful as a sub-branch of the navigation tree.
Alternative suggestion (drawing @https://docs.google.com/drawings/d/1ix0lcqFAkc6adKEfWoq0lJpakY4zIAqsdwvN8TrkOeo/edit?usp=sharing):



This is just the first layer, I'd suggest then replacing the bottom two boxes with appropriate trees, and then adding a third layer beneath that for choosing the appropriate PID repository (probably even color coded in three bands, but that's a design detail that can be decided upon later). Does that make sense?