Making Biomedical Research Software FAIR: Actionable Step-by-step Guidelines with a User-support Tool

Bhavesh Patel[1,*], Sanjay Soundarajan[1], Zicheng Hu[2]

Affiliations
1. FAIR Data Innovations Hub, California Medical Innovations Institute, San Diego, CA, USA
2. Computational Health Science, University of California San Francisco, San Francisco, CA, USA

*Corresponding author: Bhavesh Patel (bpatel@calmi2.org)

Abstract
Findable, Accessible, Interoperable, and Reusable (FAIR) guiding principles tailored for research software have been proposed by the FAIR for Research Software (FAIR4RS) Working Group. They provide a foundation for optimizing the reuse of research software. The FAIR4RS principles are, however, aspirational and do not provide practical instructions to the researchers. To fill this gap, we propose in this work the first actionable step-by-step guidelines for biomedical researchers to make their research software compliant with the FAIR4RS principles. We designate them as the FAIR Biomedical Research Software (FAIR-BioRS) guidelines. Our process for developing these guidelines, presented here, is based on a re-classification of the FAIR4RS principles and a thorough review of current practices in the field. To support researchers, we have also developed a tool that streamlines the process of implementing these guidelines. This tool is incorporated in FAIRshare, a free and open-source software application aimed at simplifying the curation and sharing of FAIR biomedical data and software through user-friendly interfaces and automation. Details about this tool are also presented.

Keywords
Code, Data curation, Metadata, Sharing, GitHub, Zenodo, Figshare

Introduction

Research software (including scripts, computational models, notebooks, code libraries, etc.) has become an increasingly important part of scientific research. A survey conducted by the Software Sustainability Institute in the UK has found that 92% of academics use research software, 69% say that their research would not be practical without it, and 56% develop their own software[1]. Other surveys have similarly shown the importance of research software in scientific research[2,3,4]. Research software plays a fundamental role not only in collecting, analyzing, and processing data but has also become the centerpiece of many scientific research projects aimed at developing computational models to understand and predict various physical phenomena. In line with this general trend, research software has also become an essential part of biomedical research over the last decade, especially with the advent of machine learning and artificial intelligence in the field. The evolution over time of the number of biomedical-related repositories created on GitHub, shown in **Fig. 1**, gives an overview of this trend.

Research software has consequently become an essential asset of scientific research and it has therefore become critical to preserve, share, and make it reusable. The Findable, Accessible, Interoperable, and Reusable (FAIR) guiding principles provide a foundation for achieving that[5]. Published in 2016, these principles are aimed at optimizing data reuse by humans and machines. While postulated for all digital research objects, several research groups have shown that the FAIR principles as written do not directly apply to software because they do not capture the specific traits of research software[6,7]. Lamprecht et al. were the first to propose in 2020 reformulated FAIR principles that are tailored to research software[6]. Noticing yet a need for more software-specific principles, the FAIR for Research Software (FAIR4RS) Working Group, jointly convened as a Research Data Alliance (RDA) Working Group, FORCE11 Working Group, and Research Software Alliance (ReSA) Task Force, released new principles in 2021 called the FAIR4RS principles[8].

Just like the original FAIR principles, the FAIR4RS principles are aspirational and aimed at providing a general framework. One study provides some actionable guidelines for making research software FAIR but it does not include a clear process and is based on the original FAIR principles rather that the FAIR4RS principles[9]. Another study also provide some actionable guidelines for making research software FAIR but it was not aimed at complying with each principle and was also based on the original FAIR principles[10]. Actionable guidelines that researchers can follow for complying with each of the FAIR4RS principles are not yet available, which is hindering the widespread adoption of FAIR practices. Particularly in biomedical research, the COVID-19 pandemic has emphasized the need for such guidelines[11]. As noted by the RDA COVID-19 Working Group: “Whilst preprints and papers are increasingly openly shared to accelerate COVID-19 responses, the software and/or source code for these papers is often not cited and hard to find, making reproducibility of this research challenging, if not impossible.”[11]

To fill this gap, we proposed in this work the first minimal and actionable step-by-step guidelines for researchers to make their research software compliant with the FAIR4RS principles. Our focus was on biomedical research software given that this effort was initiated as part of a larger project aimed at supporting the curation and sharing of COVID-19 related research data and software. We designate these guidelines as the FAIR Biomedical Research Software (FAIR-BioRS) guidelines. Many challenges have been reported to establishing such

actionable guidelines, like the lack of community agreement on metadata and identifiers[12]. Our approach consisted of deriving the best practical guidelines despite the known challenges such that researchers can already make their research software FAIR while the community is working on overcoming said challenges. The FAIR-BioRS guidelines and our approach for deriving them are presented in this manuscript.

Given that funding to support the reusability of software is lacking, the responsibility of making research software FAIR falls on the researchers developing the software. Our proposed guidelines can thus add an additional burden on these researchers who are already facing significant challenges when developing their software[13–15]. To address this concern and ensure the FAIR-BioRS guidelines are easily accessible and widely embraced by researchers, we have developed a user-support tool to streamline the process of implementing the FAIR-BioRS guidelines. This tool is incorporated in FAIRshare[16], a free and open-source desktop software application aimed at simplifying the processes for making biomedical data and software FAIR. FAIRshare takes the user's software-related dataset as input and walks them step-by-step into implementing the FAIR-BioRS guidelines through an intuitive graphical user interface and automation. Details about this tool are also provided here.

Results

Overview

Lamprecht et al. formulated the first FAIR principles tailored for research software in 2020[6]. In parallel, the FAIR4RS Working Group initiated a large-scale effort at the beginning of 2020[17] to derive FAIR principles for research software[17]. Subgroup 1 of the FAIR4RS working group was tasked to start with the original FAIR principles and propose derived principles for research software without relying on the work of Lamprecht et al. This led to the FAIR principles for research software proposed by Katz et al.[12,18] After combining the findings of all four subgroups and consulting the community[19], the FAIR4RS working group published their outcome, the FAIR4RS guiding principles, during June 2021[8]. Our work in this manuscript is based on these FAIR4RS principles given that they are the most thorough to date and backed by a large community support.

Our overall process for establishing the FAIR-BioRS guidelines is illustrated in **Fig. 2**. It consisted of classifying first the FAIR4RS principles into action-based steps. This new classification allowed us to move away from the original purpose-based classification under F, A, I, R and regroup the principles under an action-based step-by-step process instead. Next, we established for each step a checklist of items to fulfill in order to comply with all the principles classified in that step. The checklists were based on the information provided in the three studies on FAIR research software[6,8,18]. Subsequently, we conducted a thorough literature review of current practices in the field to derive actionable instructions to fulfill all the checklist items for each step, which led to the FAIR-BioRS guidelines. More details are provided in the Methods section.

New classification of the FAIR4RS principles

After reading the FAIR4RS principles, we classified them under the following actionable step-by-step process:

Step 1: Follow standards and best practices during development of the software
Step 2: Collect files to share
Step 3: Include metadata files
Step 4: Choose a license
Step 5: Share software on a repository
Step 6: Register software on a registry

While specifications on the files to share are not part of the FAIR4RS principles, discussions around what files should be shared to optimize reuse of research software are included in all three studies on FAIR for research software[6,8,18] and have therefore been included as Step 2 in our process. Details about the classification are provided in **Table 1**.

Checklists for fulfilling the FAIR4RS principles

We established for each step of the above-mentioned process a checklist of items to fulfill in order to comply with all the principles classified in that step. These checklists are presented in **Table 1**. The checklists were based on the information provided in the three studies on FAIR research software[6,8,18]. Details about the information collected from each study to establish the checklists are provided in the "fair4rsStudiesReview.csv" file included in the dataset associated with this manuscript (c.f. Data Availability section).

Actionable guidelines

Reviewed studies

Our review approach to derive practical and actionable guidelines for each step is presented in the Methods section. A list of the reviewed resources is available in the “resourcesList.csv” file included in the dataset associated with this manuscript (c.f. Data Availability section). In these resources, we looked for actionable recommendations made for any steps of the above-mentioned process. A total of 27 resources were deemed relevant to our work and included in the analysis presented next. Details about the information collected from each resource are provided in the "resourcesReview.csv" file included in the dataset associated with this manuscript (c.f. Data Availability section).

Standards and best practices

None of the reviewed resources provided actionable items for following standard development practices. This is understandable since such standards vary depending on many factors such as the domain of research and coding language. One of the reviewed resources[8] refers to the FAIRSharing Registry for a curated list of community standards[20,21].

Suggestions for best development practices are provided in 15 of the reviewed resources. Some of the reviewed resources are fully dedicated to best practices, including general best practices for scientific software development[22,23] and specific best practices for biomedical software[24,25]. Common best practice suggestions are presented in **Fig. 3A**. Overall, developing with a version control system is the most common suggestion[9–12,22–29]. GitHub, BitBucket, and GitLab are commonly mentioned ready-to-use version control repositories. The Software Sustainability Insitute’s article on “Choosing a repository for your software project” is suggested as a good guide for selecting a version control repository[30]. Using container

technologies such as Docker and Singularity[9,15,23,24,26,31], having code level documentation (in code comments, description in the headers)[22–24,27,32], and recording dependencies in a requirement.txt file or a README file[22,23] are other best practices mentioned in the reviewed resources. Best practices for documenting research software are also mentioned in 6 of the reviewed resources with one study fully dedicated to best practices for documenting scientific software[32]. Including a README file (called README, README.txt, or README.md) is the method suggested by all of the resources for maintaining documentation[10,22,23,26,32,33]. For more visibility, one of the reviewed resources[26] suggests maintaining a website using GitHub pages[34] or ReadTheDocs[35]. Several resources are suggested in the reviewed literature to help with preparing documentation for research software: Write the Docs[36], Doxygen[37], Sphinx3[38], Javadoc (Java code)[39], Roxygen (R code)[40].

Based on these findings, our recommendations for "Step 1: Follow standards and best practices during development of the software" are broken down into three sub-steps. In Step 1.a, we recommend following general standards and best practices for scientific software, such as working from a version control system (GitHub, Bitbucket, GitLab), having code level documentation (in code comments, description in the headers), and recording dependencies in a requirement.txt file or a README file. We recommend reading "Good enough practices in scientific computing"[22] and "General guidelines for biomedical software development"[24]. In Step 1.b, we recommend following language-specific standards and best practices, which depend on the development stack used. For instance, we suggest following the Python Developer's Guide[41] for Python code or Google's R Style Guide[42] for R code. In Step 1.c, we recommend following standards and best practices for documenting research software. Our suggestion is to maintain the documentation in a README.md or README.txt file located in the root directory of the software. To comply with the FAIR4RS principles (and complement the metadata discussed in the Metadata files section next), the following aspects must be documented: inputs and outputs of the software, parameters and data required to run the software, the standards applied, and how to contribute to the software. We suggest reading the paper on the "Ten simple rules for documenting scientific software"[32].

<u>Files to share</u>

From the reviewed resources, 6 have made a recommendation on the files to share for a research software (c.f. **Fig. 3B**). Sharing the source code is recommended by all of them to optimize reusability[12,18,25,28,31,43]. Providing an executable[18,28] and some input/result data[43] is also recommended.

Based on these findings, our recommendation for "Step 2: Collect files to share" is to include the source code when possible and also include executables if applicable.

<u>Metadata files</u>

From the reviewed resources, 16 have made a recommendation of metadata files and ontologies to use for research software. A majority of them[6,8–10,12,14,15,26,27,33,44,45] (c.f. **Fig. 4A**) suggest following the guidelines of the CodeMeta Project for including metadata in research software[46,47]. The CodeMeta Project is an academic-led community initiative that was the result of the FORCE 11 Software Citation working group[48]. It aims to formalize the metadata fields included in typical software metadata records and introduces important fields that did not have

clear equivalents. The CodeMeta vocabulary resulting from this effort is built over the schema.org classes SoftwareApplication and SoftwareSourceCode, which links the data for semantic web discovery. Metadata information conformant to the CodeMeta vocabulary is to be included in JSON-LD format in a file named codemeta.json and stored in the root directory of the software. The Software Heritage group has developed a CodeMeta generator that can be used to create a codemeta.json file or edit an existing one[49,50]. Tools are also available to include a codemeta.json file in R packages[51,52].

Including a citation file following the Citation File Format (CFF) is another popular suggestion amongst the reviewed resources[9,10,15,27,28,32]. The CFF was developed by a group of academics assembled under the group "Development and implementation of a standard format for CITATION files"[53]. The goal of the CFF is to provide an all-purpose citation format and specifically provide optimized means for the citation of software via the provision of software-specific reference keys and type. CFF files must be named CITATION.cff, implemented in YAML 1.2, which is a machine-readable format that optimizes human-readability, and stored in the root directory of the software[54]. The CFF file initializer is mentioned by several reviewed resources as a tool for easily generating a CITATION.cff file[55,56].

The CodeMeta and CFF are general metadata files that can be used for any research software. One of the reviewed resources[6] suggests including language-specific metadata for instance based on the DESCRIPTION file for R packages[57] or the PEP 566 for Python packages[58]. Other reviewed resources[6,26,27] also suggested preparing metadata using bio-specific ontologies such as EDAM[59–61], biotoolsSchema[62], and Bioschemas[63]. EDAM is a comprehensive ontology of well-established, familiar concepts that are prevalent within bioinformatics and computational biology[59–61]. The biotoolsSchema[62] is a formalized schema (XSD) that also includes EDAM ontology and is used by the ELIXIR Tools & Data Services Registry bio.tools[64]. Bioschemas is a community project built on top of schema.org, aiming to improve interoperability in Life Sciences so resources can better communicate and work together by using a common markup on their websites[63]. Other general ontologies suitable for software are also mentioned in the reviewed resources[6,26] such as Software Ontology[65], PROV-O[66], and RDA Research Schema[67].

Based on these findings, our recommendations for "Step 3: Include metadata files" are to include both a codemeta.json and CITATION.cff metadata files in the root directory of the software code. While these two metadata files include some overlapping fields, both are recommended since the codemeta.json file provides a machine-oriented format while the CITATION.cff provides a more human-readable format. Moreover, they both fulfill the requirement of using controlled vocabulary and include typically suggested metadata fields for FAIR software. The CodeMeta generator[49,50] and CFF file initializer[55,56] are available to help prepare both. We suggest providing all available metadata in both files. To align with the FAIR4RS principles, we recommend providing at least the following fields in the codemeta.json file:

- Software name ("name")
- Software description/abstract ("description")
- Unique identifier ("identifier")
- Authors ("givenName", "familyName") with their Organization name ("affiliation")
- Keywords ("keywords")

- Programming Language (“programmingLanguage”)
- First and current release date (“dataPublished” and “dateModified”)
- License used (“license”)

Similarly, we recommend providing at least the following fields in the CITATION.cff file:

- Authors ("given-names", “family-names") with their Organization name (“affiliation”)
- Software description/abstract (“abstract”)
- Unique identifier ("identifiers")
- Keywords (“keywords”)
- License (“license”)
- Release date (“date-released”)

License

From the reviewed resources, 14 have made suggestions about a suitable license for research software[6,8–11,15,22,24–26,28,33,44,68]. All agree that it is preferable to use an open-source license to make the software as reusable as possible (c.f. **Fig. 4B**). Since there are a large variety of open-source licenses available, it should be possible to find one that fits everyone's needs[9]. Therefore, if an open-source license is not used, that decision is expected to be properly motivated[44]. Most of the reviewed resources don’t explicitly suggest a specific license, but typically encourage choosing a license approved by the Open Source Initiative (OSI)[69] since they are well known and understood thus making reuse easier[15,26,28]. The National Institutes of Health (NIH) also suggests using an OSI-approved license[28]. Some of the reviewed resources do recommend using permissive licenses since they have very few restrictions making them optimal for reuse[6,22]. Some explicitly encourage the use of the permissive MIT and Apache 2.0 licenses[10,15,22]. The Software Package Data Exchange (SPDX)[70], Choose a License[71], and the lesson on license from the 4 Simple recommendations for Open Source Software[72] are some suggested resources for getting help with selecting a suitable license. It is typically suggested to include the license terms in a LICENSE file stored in the main directory of the software[22,24,26].

Based on these findings, our recommendation for “Step 4: Choose a license” is to include the license term in a LICENSE file located in the root directory of the software. While the FAIR4RS principles do not require research software to be open-source, we highly recommend using a license approved by the Open Source Initiative (OSI). Amongst those licenses, we encourage use of the permissive MIT[73] or Apache 2.0 licenses[10,15,74].

Repositories

From the reviewed resources, 18 made a suggestion about repositories to use for sharing research software (c.f. **Fig. 5A**). Three types of repositories are typically mentioned: general, language-specific, and domain-specific. The Registry of Research Data Repositories (r3data) is often mentioned as a tool for finding a suitable repository[75].

The general repositories that are mentioned include Zenodo, Figshare, Software Heritage, HAL (Hyper Articles on Line), Mendeley Data, and Dataverse. Zenodo[76] is the most commonly suggested repository across all reviewed studies[6,9–12,14,15,22,23,27–29,43–45,77,78]. Funded by the European Commission and developed and hosted by the Centre Européen pour la Recherche Nucléaire (CERN), Zenodo allows any digital resources, including research software, to be shared and preserved in line with the FAIR principles[79]. Zenodo assigns a separate Digital

Object Identifier (DOI) for each released version of a software and also creates a high-level DOI which refers to all versions[80]. Of the 50,000 plus DOIs registered for software, more than 80% were registered via Zenodo showing its popularity amongst the research software community[45,81]. Archiving from GitHub is made easy since it is possible to integrate GitHub and Zenodo such that each GitHub release of a software is automatically archived on Zenodo[82]. Figshare[83] is the second most recommended repository[6,10,11,15,22,25,29,43,77,78]. Figshare is a repository supported by Digital Science[84] where researchers can preserve and share any research outputs, including research software, in line with the FAIR principles[85]. Archiving GitHub repositories on Figshare is also made easy through a tool developed in collaboration with the Mozilla Science Lab[86]. Software Heritage[87] is the third most commonly suggested repository[6,9,10,12,14,27,44,45,78]. Software Heritage is an international initiative to provide a universal archive and reference system for all software[87]. It automatically and regularly harvests source code from version control systems such as GitHub. Anyone can also initiate the submission of their source code for archival[88]. Software Heritage assigns intrinsic identifiers that follow a standardized format called SoftWare Heritage persistent IDentifiers (SWHIDs). Contrary to Zenodo and Figshare, the identifiers are assigned to all artifacts over all levels of granularity such as project status, project release, state of source code, and code fragment.

Some of the reviewed resources[6,14,15,27,29,44,78] mention language-specific repositories such as PyPI[89] and Conda[90] for Python packages, CRAN for R packages[91], and Dockstore for Docker-based tools[92]. Only two biomedical-specific repositories were mentioned in the reviewed studies[6,28,43]: ModelDB for computational neuroscience models[93,94] and Bioconductors for R-packages aimed at the analysis of genomics data[95–97].

Our recommendations for “Step 5: Share software on a repository” are two-fold. In Step 5.a, we suggest sharing on a language-specific repository (e.g., PyPI[89] or Conda[90] for Python packages, CRAN for R packages[91], Dockstore for Docker-based tools[92]) or a biomedical-specific repository (e.g., ModelDB for computational neuroscience models[93,94] and Bioconductors for R-packages aimed at the analysis of genomics data[95–97]) if applicable. In Step 5.b., we suggest sharing on Zenodo or Figshare. This step must be completed even if Step 5.a is followed since none of the repositories mentioned in Step 5.a have provided a clear assessment of their compliance with FAIR principles contrary to Zenodo[79] and Figshare[85]. Software Heritage is another option and has the advantage of automatically assigning a unique identifier for all levels of granularity (as recommended in the FAIR4RS principle F1.1) but we could not find an assessment of compliance with other FAIR principles.

<u>Registries</u>

From the reviewed resources, 5 made a suggestion about registries to use for research software (c.f. **Fig. 5B**)[6,12,14,26,27]. The repositories Zenodo, Figshare, Software Heritage, CRAN, PyPI, and Conda are also mentioned as registries in the reviewed literature since they require specific metadata that is used for indexing. bio.tools[98] is the most commonly suggested registry. Supported by ELIXIR, the European Infrastructure for Biological Information, bio.tools provides a registry of biomedical tools[64] and contains more than 24,000 registered tools (as of April 2022). As explained in the metadata files section above, bio.tools uses the biotoolsSchema, including EDAM ontology, to describe registered bioinformatics software. Another possibility, not found in the reviewed resources but known to the authors, is to register software on the

Research Resource Identifiers (RRIDs) portal and obtain an RRID[99]. RRIDs are unique numbers assigned to help researchers cite key resources, including software projects, in the biomedical literature to improve transparency of research methods[100].

Our recommendation for "Step 6: Share software on a registry" is to optionally register the software on bio.tools. Even if Zenodo and Figshare (Step 5.b) already act as registries, it is still suggested to register on bio.tools to increase findability and create additional rich metadata about the software in a standard format and ontology other than those followed by codemeta.json and CITATION.cff. Alternatively, software can be registered on the RRIDs portal[101].

FAIR Biomedical Research Software guidelines

Based on the above-mentioned findings from our review, we derived the FAIR Biomedical Research Software (FAIR-BioRS) guidelines. These guidelines are presented in **Table 2**. Implementing these guidelines will ensure compliance with the FAIR4RS principles as shown in the assessment presented in **Table 3**. These guidelines and the compliance assessment are maintained in a GitHub repository[102]. This repository will serve as a tool to collect community feedback on the FAIR-BioRS guidelines. New versions of the guidelines, that will be established based on community feedback or evolution of current standards, will be maintained there. Each version of the guidelines and associated compliance assessment will be saved as a GitHub release and simultaneously archived on Zenodo such that it is preserved and uniquely identifiable. The current version v1.0.1 of the guidelines associated with this manuscript is similarly archived on Zenodo[103].

FAIRshare

Guidelines typically have little effect and are very unlikely to be adopted without proper tools to support them. Tools and resources are available to implement some steps of the FAIR-BioRS guidelines[49,56,71,104] but they are dispersed and not logically connected to streamline the process. Therefore, we developed and integrated a process to implement the FAIR-BioRS guidelines in our software application called FAIRshare. FAIRshare is aimed at streamlining the processes for making biomedical research data and software FAIR[16]. Specifically, FAIRshare combines intuitive user interfaces, automation, and user support resources into a single location to guide and assist the researchers through a suitable process for making their research data and software FAIR and sharing it on an adequate repository. FAIRshare is free and developed under the permissible open-source MIT license to encourage community contributions for updating existing workflows and including new ones. Details about the development of FAIRshare are provided in the Methods section. The current version of FAIRshare (v.1.4.0) and its documentation (v3.0.0)[105] can be accessed through their respective GitHub repositories[16,106], where their source code is also available.

The process to make research software FAIR with FAIRshare guides users through the steps of the FAIR-BioRS guidelines in a slightly different order to optimize the process, as illustrated in **Fig. 7**. In step 1, FAIRshare asks the user to select the location of their software-related files. FAIRshare provides users with the ability to select files from either their computer or one of their GitHub repositories. If the user selects GitHub, FAIRshare provides two options to connect FAIRshare to their GitHub account: 1) Using their username and password,

or 2) Using a personal access token. In step 2, FAIRshare asks a series of “Yes/No” questions to verify that the user has followed applicable standards and best practices during development of the software as dictated by Step 1 of the FAIR-BioRS guidelines. Links to the relevant resources on standards and best practices suggested by the FAIR-BioRS guidelines are provided in the interface. In step 3, FAIRshare provides a convenient form-type interface for the user to provide information about their software. The fields of the form correspond to fields from the codemeta.json and CITATION.cff metadata files and the user’s entries are used to automatically generate and include these metadata files in the user’s high-level dataset directory. If a codemeta.json file is found at the location selected by the user in step 1, information from that file is automatically pulled by FAIRshare to pre-populate the form. If no codemeda.json file is found and the user files are located in a GitHub repository, FAIRshare will automatically pull available information from the repository metadata (authors, keywords, etc.) and pre-populated related fields of the form. Throughout the forms, tooltips have been included to help users enter the required information. Suggestions are also made for some of the fields to help the user provide inputs rapidly (e.g., “National Institutes of Health” is suggested for the “Funder” field, “Scientific” is suggested for the “Software type” field, etc.). If any of the fields require an entry following a controlled vocabulary for the codemeta.json or CITATION.cff metadata files, FAIRshare typically provides a dropdown list of standardized options to select from to ensure metadata files are generated without error. FAIRshare also makes it mandatory to provide inputs for the metadata fields that were deemed essential by the FAIR-BioRS guidelines. In step 4, the user is prompted to select a license for their software. A dropdown list is provided to select one of the licenses recommended by the OSI. The “MIT” and “Apache 2.0” licenses are suggested in the user interface since they have been deemed optimal for making research software FAIR. The user can read the terms of a selected license directly in the app and can request FAIRshare to include a LICENSE file with associated terms in their software. In step 5, the user is prompted to select a repository to share their software files. In the current release, FAIRshare supports sharing on Zenodo if the software-related files are on GitHub and supports sharing of Zenodo and Figshare if the software-related files are on the user’s computer. A convenient interface is provided to connect FAIRshare to the user’s Zenodo or Figshare account using a token. Upon login, the user is prompted to provide information about their software that is required by the selected repository. FAIRshare provides a form-type interface such that the user can easily input that information. The interface closely mimics the interface from the selected repository platform to maintain familiarity. Since most of the information required by Zenodo and Figshare is similar to that required in the codemeta.json file, FAIRshare pre-populates the form based on user inputs during step 3. The subsequent process depends on the location of the user’s files as specified during step 1. If the files are located on the user’s computer, a summary of the user's dataset files, including files to be generated by FAIRshare, is provided to the user in step 6. Once the user hits the “Start upload” button, FAIRshare generates the codemeta.json and CITATION metadata files and LICENSE file in the dataset folder, zips the dataset folder, and upload it in a draft deposition on the selected repository. Upon completion, the user is prompted to publish their dataset on the repository in step 7, such that it becomes publicly accessible. If the files are located on a GitHub repository, FAIRshare makes use of the existing connection between Zenodo and GitHub to facilitate archiving of the user’s software files from GitHub to Zenodo[104]. In step 6, instructions are

provided to the user to go on the Zenodo webpage and connect their GitHub repository to Zenodo. Once this connection is created, everytime a release will be published on the GitHub repository, it will be automatically published on Zenodo as well. Once FAIRshare detects that the connection has been established, the user is prompted to click on a “Start upload” button to let FAIRshare generate and push the codemeta.json and CITATION metadata files, LICENSE file, and .zenodo.json file on their GitHub repository. After the upload is completed, the user is guided in step 7 to publish a release of their repository on GitHub, which will automatically push a release on Zenodo as well using the information from the .zenodo.json file. In both cases, a message is shown to the user after completing step 7 to encourage them to register their software on bio.tools. If the user selects to do so, an interface is provided in step 8 where the user can connect FAIRshare to their bio.tools account, enter basic metadata required by bio.tools, and register their software.

Discussion

While research software constitutes the backbone of biomedical research, the amount of effort dedicated to ensuring software reusability and long-term sustainability is nowhere near the effort dedicated to data. The Findable, Accessible, Interoperable, and Reusable (FAIR) guiding principles published provide a foundation for managing digital research objects, including software, such that their reusability by humans and machines is optimized. However, they fail to capture the specific traits of software such as dependencies and versioning. The Research Data Alliance (RDA) FAIR for Research Software (FAIR4RS) Working Group derived much-needed reformulated FAIR principles to address this shortcoming. Just like the original FAIR guiding principles, the FAIR4RS guiding principles are still aspirational and do not provide practical instructions and actionable items to the researchers since they may be domain-specific while the principles are meant to cover all research software. To fill this gap, we proposed in this work the first minimal and actionable step-by-step guidelines for biomedical researchers to make their research software compliant with the FAIR4RS principles. Our process for deriving these guidelines started by reclassifying the FAIR4RS principles into action-based steps. Then, we establish a checklist for each of these steps to fulfill the requirements of all the principles classified in that step. Subsequently, we conducted a thorough literature review to find out available practices to fulfill each item on our checklist. Through analysis of our findings, we established the FAIR-BioRS guidelines, which are the first minimal and actional guidelines for making research software FAIR by fulfilling all the requirements of the FAIR4RS principles. The reviewed resources included the best practices suggested by the NIH for sharing research software[28]. Therefore, the FAIR-BioRS guidelines align with the recommendations of the NIH for sharing research software. To ensure the FAIR-BioRS guidelines are easily accessible and widely embraced by researchers, we have developed a user-support tool to streamline the process of implementing the FAIR-BioRS guidelines. This tool is incorporated in FAIRshare, a software application aimed at simplifying the processes for making biomedical research data and software FAIR. FAIRshare takes the user’s software-related dataset as input and walks them step-by-step into implementing the FAIR-BioRS guidelines through an intuitive graphical user interface and automation. This way, researchers can implement the FAIR-BioRS guidelines even without prior knowledge of them and in fact learn about these guidelines along the way. FAIRshare is free and open-source to encourage community contributions for keeping up with

the ever-evolving FAIR guidelines. We expect the FAIR-BioRS guidelines to evolve over time based on suggestions and feedback from the research software community as well as development of new standards. We have thus established a dedicated GitHub repository where new versions of the the FAIR-BioRS guidelines will be maintained and published[102]. This repository will also be used to receive community feedback and suggestions through the GitHub issues.

We expect next to establish more specific guidelines for different types of software (e.g., establish the FAIR-BioRS Python package guidelines or the FAIR-BioRS Jupyter notebook guidelines). In parallel, we plan to integrate additional resources in FAIRshare such as sharing from GitHub to Figshare or registering on the RRID portal. We also plan to include additional automation to further streamline the FAIRshare process for making research software FAIR. These include for instance automatically validating implementation of language-specific standards or prefilling keywords and descriptions using Natural Language Processing. As the guidelines, FAIRshare, and other resources related the FAIR-BioRS guidelines are expected to evolve over time, we have established a dedicated GitHub repository called the FAIR-BioRS Hub[102]. We refer to that repository to access the latest versions of all things related to the FAIR-BioRS guidelines and track the interaction between their different versions.

With the availability of clearly defined actionable guidelines and an associated tool to streamline the process, we are hoping the biomedical research software community will embrace the FAIR practices which will ultimately increase the pace of discoveries and innovations.

Methods

Definition of research software

The FAIR4RS Working Group was constituted of four subgroups, each focusing on different aspects of their effort[107]. Group 3 focused on defining research software through a thorough review of the literature. In the outcome of their effort, they provided the following short and concise definition: “Research Software includes source code files, algorithms, scripts, computational workflows and executables that were created during the research process or for a research purpose. Software components (e.g., operating systems, libraries, dependencies, packages, scripts, etc.) that are used for research but were not created during or with a clear research intent should be considered software in research and not Research Software”[108]. The actionable guidelines developed in this work are for research software based on this definition. This includes code, scripts, models, notebooks, libraries, executables, and other forms as long as it fits the previously cited definition. Given that our work was born with the aim of making COVID-19-related data and research software FAIR, our focus was on deriving guidelines for biomedical research software, although some of the findings may be applicable to other fields of research.

Re-classification of the FAIR4RS principles

The FAIR4RS principles contain 17 guiding principles for FAIR research software (vs 15 in the original FAIR principles), including 6 for Findability, 4 for Accessibility, 2 for Interoperability, and 5 for Reusability. After analyzing the FAIR4RS guiding principles, we classified the principles

into the action-based steps shown in **Table 1**. Subsequently, we re-read the details provided for the principles by the FAIR4RS working group and formulated an actionable instruction for each. We repeated this process for the Lamprecht et. al.[6] and Katz et al.[18] studies since they have overlapping principles with the FAIR4RS working group. Eventually, we combined the actionable instructions collected from the three studies and defined a single actionable instruction for each principle. Details about this process are included in the "fair4rsStudiesReview.csv" file included in the dataset associated with this manuscript (c.f. Data Availability section). Finally, we combined the actionable instructions from each principle within a step and proposed a checklist of items to fulfill for complying with all the principles classified within that step (**Table 1**).

Literature review for practical implementation of FAIR4RS principles

Our next step was to derive actionable guidelines for fulfilling these checklists, i.e., provide a response to the question "How do I practically make my research software FAIR?". We conducted a literature review for identifying actionable guidelines for each item on the checklists. Since discussion around FAIR for research software is very recent and not widespread in the literature, we reviewed literature not only related to FAIR research software, but also related to making research software reusable without necessarily a reference to the FAIR principles. Our review was also not restricted to biomedical-related articles for the same reason. Our overall review strategy consisted of including resources from the following categories:

- Category 1: We started by reviewing the three studies on FAIR principles for research software[6,8,18].
- Category 2: Subsequently, we read fully the resources referenced in studies from category 1 that were deemed relevant based on their title and how they were discussed in category 1 studies.
- Category 3: We also reviewed resources listed in the FAIR4Software reading materials[109], FAIR4RS Subgroup 4 reading list of new research[110], and the Zenodo community page of the FAIR4RS Working Group[111] that were deemed relevant based on their title and abstract and were not already encountered during the previous steps.
- Category 4: Given our focus on biomedical research software, we also searched literature related to FAIR for research software on PubMed. The start period was set to 2015 since the concept of FAIR data was introduced then. We read resources that were deemed relevant based on their title and abstract and were not already encountered during the previous steps.
- Category 5: Finally, we included additional studies from the authors' knowledge that were not already read during the previous steps.

Details about the review process are provided in the PRISMA diagram in **Fig. 6**. More details about the review strategy are included in the "searchStrategy.csv" file associated with this manuscript (c.f. Data Availability section). A list of all the resources encountered during our review is provided in the "resourcesList.csv" file associated with this manuscript (c.f. Data Availability section). In line with our step-by-step process and checklist from **Table 1**, we decided to collect information pertinent to the following topics from all reviewed resources:

1. Standards/best practices suggested to be followed and resources mentioned to help with standards/best practices

2. Documentation or elements of documentation suggested to be provided and resources mentioned to help with documentation
3. Software files suggested to be shared and resources mentioned for helping with these files
4. Metadata files suggested to be included and resources to help with metadata
5. Licenses suggested to be used and resources mentioned to help with license
6. Repositories suggested for sharing research software and resources mentioned to help with repository selection
7. Registries suggested for software and resources mentioned to help with registry selection

Initially, we copied word-for-word the relevant information as written in the reviewed resources. This is available in the "resourcesReview.csv" file associated with this dataset (c.f. Data Availability section). Subsequently, we only retained keywords such as the name of the suggested metadata file, repository, etc. to facilitate further analysis. Information from any of the topics that applied to only specific fields of research other than biomedical were left out. This is available in the "resourcesReviewKeywords.csv" file associated with this dataset (c.f. Data Availability section)

We looked at the frequency of the keywords in each topic across the resources reviewed This was achieved by developing a Jupyter notebook-based script (c.f. Code Availability section) that used scientific packages such as pandas[112,113], Matplotlib[114,115], and seaborn[116]. Through analysis of these results and the recommendations read in the reviewed studies, we derived minimal and actionable step-by-step guidelines for making research software FAIR, i.e. the FAIR-BioRS guidelines, such that all the requirements of the FAIR4RS principles are met. For each step, we also provided relevant user support resources that were identified in our review.

User support tool: FAIRshare

FAIRshare is inspired by the software called SODA for SPARC, which has been developed by our team to assist researchers funded by the NIH SPARC (Stimulating Peripheral Activity to Relieve Conditions) program in making their data FAIR according to the SPARC data standards[117–119]. Specifically, FAIRshare combines intuitive user interfaces and automation to guide and assist the researchers through the suitable process for making their research data and software FAIR and sharing it on an adequate repository. FAIRshare is developed as a cross-platform desktop software application, such that the data of the researchers remain on their computer or a storage medium of their choice until they are ready to share it on a suitable repository. It is built using Electron, GitHub's framework for building cross-platform desktop applications using web technologies (HTML, CSS, JavaScript, NodeJS)[120]. Vue3 is used as the frontend framework to build intuitive, interactive, and responsive user interfaces[121]. The backend is built using Flask, a micro web framework written in Python[122]. The use of Python for the backend of the application was motivated by the popularity of Python in the biomedical research field and the availability of relevant existing packages and APIs for curating research data and software. More details are available in the dedicated GitHub repository for FAIRshare[16] (c.f. Code Availability section).

Data Availability

The data associated with this manuscript consist of several csv files that contain details about data collected and processed during the review portion of this manuscript. Since no FAIR guidelines were found for review data, we structured the dataset according to the SPARC Data Structure (SDS), which provides a broad data and metadata structure to organize biomedical research data according to the FAIR principles[118]. The SPARC data curation software SODA for SPARC[117–119] was used to organize the data and prepare the metadata files. The dataset was maintain from a GitHub repository[123] and the latest version associated with this manuscript (v2.0.0) is also archived on Zenodo[124] with the help of FAIRshare and under the permissible Creative Commons Attribution 4.0 International (CC-BY) license[125].

Code Availability

The code associated with this manuscript consists of a “main.ipynb” Jupyter notebook, the source code of FAIRshare, and the source code of the FAIRshare documentation. The “main.ipynb” Jupyter notebook contains the code used to analyze the findings from the “resourcesReviewKeywords.csv” and to conduct other analysis presented in this manuscript (e.g., generate **Fig. 1**). This notebook was developed with GitHub as source control[126]. The dataset associated with this notebook was made FAIR according to the FAIR-BioRS guidelines using FAIRshare, and shared under the permissible MIT license. The latest version associated with this manuscript (v2.0.0) is archived on Zenodo[127]. The source code for FAIRshare is also hosted on GitHub[16]. The current version of FAIRshare (v1.4.0) discussed in this manuscript was made FAIR using FAIRshare itself, and shared on Zenodo[128] under the permissible MIT license. The source code for the FAIRshare documentation is hosted on GitHub as well[129] and the current version (3.0.0) was shared under the permissible MIT license on Zenodo[130].

Acknowledgements

This work was supported by grants for the NIH U01AI150741 and NIH SPARC OT2OD030213.

Author contributions

B. Patel led the conception and design of the FAIR-BioRS guidelines, managed the acquisition/analysis/interpretation of data from the reviewed studies, supervised the development of FAIRshare, and contributed to drafing/revising the manuscript. S. Soundarajan lead the development of FAIRshare and contributed to revising the manuscript. H. Zicheng contributed to the conception and design of the FAIR-BioRS guidelines, and revising the manuscript.

Competing interests

The authors declare no competing interests.

References

1. Hettrick, S. *softwaresaved/software_in_research_survey_2014: Software in research survey*. (2018). doi:10.5281/zenodo.1183562.
2. Nangia, U. & Katz, D. S. Track 1 Paper: Surveying the U.S. National Postdoctoral Association Regarding Software Use and Training in Research. (2017) doi:10.6084/m9.figshare.5328442.v1.
3. Hannay, J. E. *et al.* How do scientists develop and use scientific software? in *2009 ICSE Workshop on Software Engineering for Computational Science and Engineering* 1–8 (2009).
4. Prabhu, P. *et al.* A survey of the practice of computational science. in *SC '11: State of the Practice Reports* 1–12 (IEEE, 2011).
5. Wilkinson, M. D. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Sci Data* **3**, 160018 (2016).
6. Lamprecht, A.-L. *et al.* Towards FAIR principles for research software. *Data sci.* **3**, 37–59 (2020).
7. Katz, D. S. *et al.* Software vs. data in the context of citation. *PeerJ Preprints* **4**, (2016).
8. Chue Hong, N. P. *et al.* FAIR Principles for Research Software (FAIR4RS Principles). *Research Data Alliance* (2021) doi:10.15497/RDA00065.
9. Hasselbring, W., Carr, L., Hettrick, S., Packer, H. & Tiropanis, T. From FAIR research data toward FAIR and open research software. *it - Information Technology* **62**, 39–47 (2020).
10. Martinez-Ortiz, C., Kuzak, M., Spaaks, J. H., Maassen, J. & Bakker, T. *Five recommendations for 'FAIR software'*. (2020). doi:10.5281/zenodo.4310217.
11. RDA COVID-19 Working Group. RDA COVID-19 Recommendations and Guidelines on Data Sharing. *Research Data Alliance* (2020) doi:10.15497/rda00052.
12. Katz, D. S. *et al.* A Fresh Look at FAIR for Research Software. *arXiv [cs.SE]* (2021).

13. Peer, L. *et al.* Challenges of Curating for Reproducible and FAIR Research Output. (2021) doi:10.15497/RDA00063.
14. Gruenpeter, M. *et al.* M2.15 Assessment report on 'FAIRness of software'. (2020) doi:10.5281/zenodo.4095092.
15. Anzt, H. *et al.* An environment for sustainable research software in Germany and beyond: current state, open challenges, and call for action. *F1000Res.* **9**, 295 (2021).
16. FAIRshare: Simplifying the curation and sharing of biomedical research data and software according to applicable FAIR guidelines. *GitHub* https://github.com/fairdataihub/FAIRshare.
17. FAIR for Research Software (FAIR4RS) WG. https://rd-alliance.org/groups/fair-research-software-fair4rs-wg (2020).
18. Katz, D. S., Gruenpeter, M. & Honeyman, T. Taking a fresh look at FAIR for research software. *Patterns (N Y)* **2**, 100222 (2021).
19. Katz, D. S., Chue Hong, N. P., Barker, M. & Gruenpeter, M. *FAIR4RS WG subgroup community consultation March 2021*. (2021). doi:10.5281/zenodo.4635410.
20. FAIRsharing - standards. *FAIRsharing* https://fairsharing.org/standards/.
21. Sansone, S.-A. *et al.* FAIRsharing as a community approach to standards, repositories and policies. *Nat. Biotechnol.* **37**, 358–367 (2019).
22. Wilson, G. *et al.* Good enough practices in scientific computing. *PLoS Comput. Biol.* **13**, e1005510 (2017).
23. Code Refinery — Reproducible research documentation. https://coderefinery.github.io/reproducible-research/.
24. Silva, L. B., Jimenez, R. C., Blomberg, N. & Oliveira, J. L. General guidelines for biomedical software development. *F1000Research* vol. 6 273 (2017).
25. Leprevost, F. da V., Barbosa, V. C., Francisco, E. L., Perez-Riverol, Y. & Carvalho, P. C. On best practices in the development of bioinformatics software. *Front. Genet.* **5**, 199 (2014).
26. Jiménez, R. C. *et al.* Four simple recommendations to encourage best practices in research

software. *F1000Res.* **6**, (2017).

27. Erdmann, C. *et al. Top 10 FAIR Data & Software Things*. (2019). doi:10.5281/zenodo.2555498.
28. Best Practices for Sharing Research Software. *NIH* https://datascience.nih.gov/tools-and-analytics/best-practices-for-sharing-research-software-faq.
29. Ayers, P. LibGuides: Citing & publishing software: Publishing research software. (2016).
30. Hong, N. C. Choosing a repository for your software project. *Software Sustainability Institute* https://software.ac.uk/choosing-repository-your-software-project.
31. Madduri, R. *et al.* Reproducible big data science: A case study in continuous FAIRness. *PLoS One* **14**, e0213013 (2019).
32. Lee, B. D. Ten simple rules for documenting scientific software. *PLoS Comput. Biol.* **14**, e1006561 (2018).
33. Di Cosmo, R. *et al.* Curated archiving of research software artifacts: lessons learned from the French open archive (HAL). in *IDCC 2020-International Digital Curation Conference* (2020). doi:10.2218/ijdc.v15i1.698.
34. GitHub Pages. https://pages.github.com/.
35. Read the Docs. https://readthedocs.org/.
36. A beginner's guide to writing documentation — Write the Docs. https://www.writethedocs.org/guide/writing/beginners-guide-to-docs/.
37. Doxygen: Doxygen. https://www.doxygen.nl/index.html.
38. Overview — sphinx documentation. https://www.sphinx-doc.org/en/master/.
39. Javadoc tool home page. https://www.oracle.com/java/technologies/javase/javadoc-tool.html.
40. Roxygen2. https://roxygen2.r-lib.org/.
41. Python Developer's Guide. *Python.org* https://www.python.org/dev/.

42. Google's R style guide. https://google.github.io/styleguide/Rguide.html.
43. Benureau, F. C. Y. & Rougier, N. P. Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions. *Front. Neuroinform.* **11**, 69 (2017).
44. European Commission, Directorate-General for Research and Innovation. Scholarly Infrastructures for Research Software: Report from the EOSC Executive Board Working Group (WG) Architecture Task Force (TF) SIRS. *Publications Office* (2020) doi:10.2777/28598.
45. Ferguson, C. *et al. D3.1 Survey of Current PID Services Landscape*. (2018). doi:10.5281/zenodo.1324296.
46. codemeta: Minimal metadata schemas for science software and code, in JSON-LD. *GitHub* https://github.com/codemeta/codemeta.
47. codemeta. The CodeMeta project. https://codemeta.github.io/.
48. force11-sciwg: FORCE11 Software Citation Implementation Working Group. *GitHub* https://github.com/force11/force11-sciwg.
49. CodeMeta generator. https://codemeta.github.io/codemeta-generator/.
50. codemeta-generator. *GitHub* https://github.com/codemeta/codemeta-generator.
51. codemetar: an R package for generating and working with codemeta. *GitHub* https://github.com/ropensci/codemetar.
52. Boettiger, C. *et al. ropensci/codemetar: codemetar 0.3.0*. (2021). doi:10.5281/zenodo.4748266.
53. Druskat, S. *Citation File Format (CFF)*. (2017). doi:10.5281/zenodo.1117789.
54. YAML Ain't Markup Language (YAML™) revision 1.2.2. https://yaml.org/spec/1.2.2/.
55. cffinit: a web form to initialize CITATION.cff files. https://citation-file-format.github.io/cff-initializer-javascript/#/.
56. cff-initializer-javascript: Web form to initialize CITATION.cff files. *GitHub* https://github.com/citation-file-format/cff-initializer-javascript.

57. Wickham, H. & Bryan, J. Package metadata. https://r-pkgs.org/description.html.
58. PEP 566 – metadata for python software packages 2.1. https://peps.python.org/pep-0566/.
59. EDAM: Ontology of bioscientific data analysis and data management. http://edamontology.org/.
60. Ison, J. *et al.* EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats. *Bioinformatics* **29**, 1325–1332 (2013).
61. Ison, J. *et al. edamontology/edamontology: EDAM 1.25*. (2020). doi:10.5281/zenodo.3899895.
62. biotoolsSchema: biotoolsSchema : Tool description data model for computational tools in life sciences. *GitHub* https://github.com/bio-tools/biotoolsSchema.
63. Bioschemas - 1.0 release. https://bioschemas.org/profiles/ComputationalTool/1.0-RELEASE.
64. Ison, J. *et al.* The bio.tools registry of software tools and data resources for the life sciences. *Genome Biol.* **20**, 164 (2019).
65. The Software Ontology - homepage. http://theswo.sourceforge.net/.
66. PROV-O: The PROV ontology. https://www.w3.org/TR/prov-o/.
67. Research metadata schemas WG. *RDA* https://www.rd-alliance.org/groups/research-metadata-schemas-wg (2019).
68. Katz, D. S. *et al.* Recognizing the value of software: a software citation guide. *F1000Res.* **9**, 1257 (2020).
69. Licenses & standards. https://opensource.org/licenses.
70. International open standard (ISO/IEC 5962:2021) - software package data exchange (SPDX). *Software Package Data Exchange (SPDX)* https://spdx.dev/ (2021).
71. Choose an open source license. *Choose a License* https://choosealicense.com/.
72. Adopt a licence and comply with the licence of third-party dependencies. https://softdev4research.github.io/4OSS-lesson/03-use-license/index.html.

73. The MIT license. https://opensource.org/licenses/MIT.

74. Apache license, version 2.0. https://opensource.org/licenses/Apache-2.0.

75. Registry of Research Data Repositories. https://www.re3data.org/.

76. Zenodo. https://zenodo.org/.

77. Smith, A. M., Katz, D. S. & Niemeyer, K. E. Software citation principles. *PeerJ Comput. Sci.* **2**, e86 (2016).

78. Jackson, M. *Software Deposit: Where to deposit software*. (2018). doi:10.5281/zenodo.1327329.

79. Zenodo-Principles. https://about.zenodo.org/principles/.

80. Rix, K. Expert evidence: Frequently asked questions. *J. Forensic Leg. Med.* **77**, 102106 (2021).

81. Fenner, M. DOI Registrations for Software. (2018) doi:10.53731/r7ae36h-97aq74v-ag5pw.

82. Referencing and citing content - GitHub. https://docs.github.com/en/repositories/archiving-a-github-repository/referencing-and-citing-content.

83. Figshare - credit for all your research. https://figshare.com/.

84. Figshare. *Digital Science* https://www.digital-science.com/product/figshare/ (2020).

85. Splawa-Neyman, P. Figshare and the FAIR data principles. (2018) doi:10.6084/m9.figshare.7476428.v1.

86. Code as a Research Object. http://mozillascience.github.io/code-research-object/.

87. Software Heritage. https://www.softwareheritage.org//.

88. Software Heritage Archive. https://archive.softwareheritage.org/.

89. PyPI · the Python Package Index. *PyPI* https://pypi.org/.

90. Conda — Conda documentation. https://docs.conda.io/en/latest/.

91. Hornik, K. The comprehensive R archive network. *Wiley Interdiscip. Rev. Comput. Stat.* **4**, 394–398 (2012).

92. Dockstore. https://dockstore.org/.

93. ModelDB: Home. https://senselab.med.yale.edu/ModelDB/.

94. McDougal, R. A. *et al.* Twenty years of ModelDB and beyond: building essential modeling tools for the future of neuroscience. *J. Comput. Neurosci.* **42**, 1–10 (2017).

95. Bioconductor - home. https://www.bioconductor.org/.

96. Huber, W. *et al.* Orchestrating high-throughput genomic analysis with Bioconductor. *Nat. Methods* **12**, 115–121 (2015).

97. Gentleman, R. C. *et al.* Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.* **5**, R80 (2004).

98. bio.tools · Bioinformatics Tools and Services Discovery. https://bio.tools/.

99. Research Resource Identifiers. https://www.rrids.org/.

100. Bandrowski, A. *et al.* The Resource Identification Initiative: A Cultural Shift in Publishing. *Neuroinformatics* **14**, 169–182 (2016).

101. Research resource identifier. *Research Resource Identifier* https://www.rrids.org/.

102. FAIR-BioRS-guidelines: Repository of the FAIR Biomedical Research Software (FAIR-BioRS) guidelines. *GitHub* https://github.com/fairdataihub/FAIR-BioRS.

103. Patel, B. & Soundarajan, S. FAIR Biomedical Research Software (FAIR-BioRS) guidelines. (2022) doi:10.5281/zenodo.6609011.

104. Foster, E. How to make your code citable. *Library Guides* https://guides.lib.berkeley.edu/citeyourcode (2018).

105. FAIRshare documentation website. https://docs.fairshareapp.io/.

106. FAIRshare-Docs: Documentation for the FAIRshare application. *GitHub* https://github.com/fairdataihub/FAIRshare-Docs.

107. FAIR for Research Software (FAIR4RS) WG. *RDA* https://www.rd-alliance.org/groups/fair-research-software-fair4rs-wg (2020).

108. Gruenpeter, M. *et al. Defining Research Software: a controversial discussion*.

https://docs.google.com/document/d/139vi8KCz2h0KyYfhN46SR7bEuJ3nggYgb1kaN6CNkSQ/edit# (2021).

109. FAIR4Software reading materials. https://www.rd-alliance.org/group/software-source-code-ig/wiki/fair4software-reading-materials (2020).

110. FAIR4RS Working Group. *FAIR4RS Subgroup 4 - reading list of new research*. (2021). doi:10.5281/zenodo.4555865.

111. FAIR for Research Software (FAIR4RS) Zenodo Community. https://zenodo.org/communities/fair4rs.

112. McKinney, W. Data Structures for Statistical Computing in Python. in *Proceedings of the 9th Python in Science Conference* (SciPy, 2010). doi:10.25080/majora-92bf1922-00a.

113. The pandas development team. *pandas-dev/pandas: Pandas 1.4.2*. (2022). doi:10.5281/zenodo.6408044.

114. Hunter, J. D. Matplotlib: A 2D Graphics Environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).

115. Caswell, T. A. *et al. matplotlib/matplotlib: REL: v3.5.2*. (2022). doi:10.5281/zenodo.6513224.

116. Waskom, M. seaborn: statistical data visualization. *J. Open Source Softw.* **6**, 3021 (2021).

117. Patel, B., Srivastava, H., Aghasafari, P. & Helmer, K. SPARC: SODA, an interactive software for curating SPARC datasets. *FASEB J.* **34**, 1–1 (2020).

118. Bandrowski, A. *et al.* SPARC Data Structure: Rationale and Design of a FAIR Standard for Biomedical Research Data. *bioRxiv* 2021.02.10.430563 (2021) doi:10.1101/2021.02.10.430563.

119. Patel, B. *et al. SODA (Software to Organize Data Automatically) for SPARC*. (2022). doi:10.5281/zenodo.6426344.

120. Electron. https://www.electronjs.org/.

121. Vue.Js - the progressive JavaScript framework. https://vuejs.org.

122. Flask documentation (2.1.X). https://flask.palletsprojects.com/en/2.1.x/.

123. *FAIR-BioRS-data*. (Github).

124. Patel, B., Soundarajan, S. & Hu, Z. Dataset: FAIR Biomedical Research Software (FAIR-BioRS) manuscript. (2022) doi:10.5281/zenodo.6604793.

125. Creative commons — attribution 4.0 international — CC BY 4.0. https://creativecommons.org/licenses/by/4.0/.

126. FAIR-BioRS-code. *GitHub* https://github.com/fairdataihub/FAIR-BioRS-code.

127. Patel, B. *Code: FAIR Biomedical Research Software (FAIR-BioRS) manuscript*. (2022). doi:10.5281/zenodo.6468964.

128. Soundarajan, S. & Patel, B. *FAIRshare: FAIR data and software sharing made easy*. (2022). doi:10.5281/zenodo.6612672.

129. FAIRshare-Docs: Documentation for the FAIRshare application. *GitHub* https://github.com/fairdataihub/FAIRshare-Docs.

130. Soundarajan, S. & Patel, B. FAIRshare docs. (2022) doi:10.5281/zenodo.6612747.

131. Druskat, S., Haines, R. & Baker, J. *Citation File Format (CFF) - Specifications*. (2018). doi:10.5281/zenodo.1242911.

132. Licenses & standards. https://opensource.org/licenses.

Figures

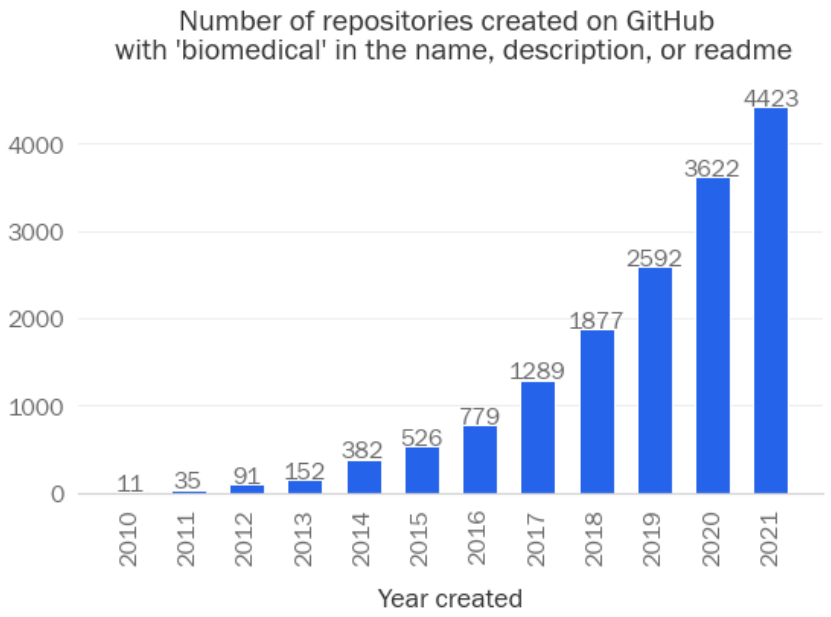

Figure 1. Evolution of the number of biomedical-related repositories on GitHub over time.

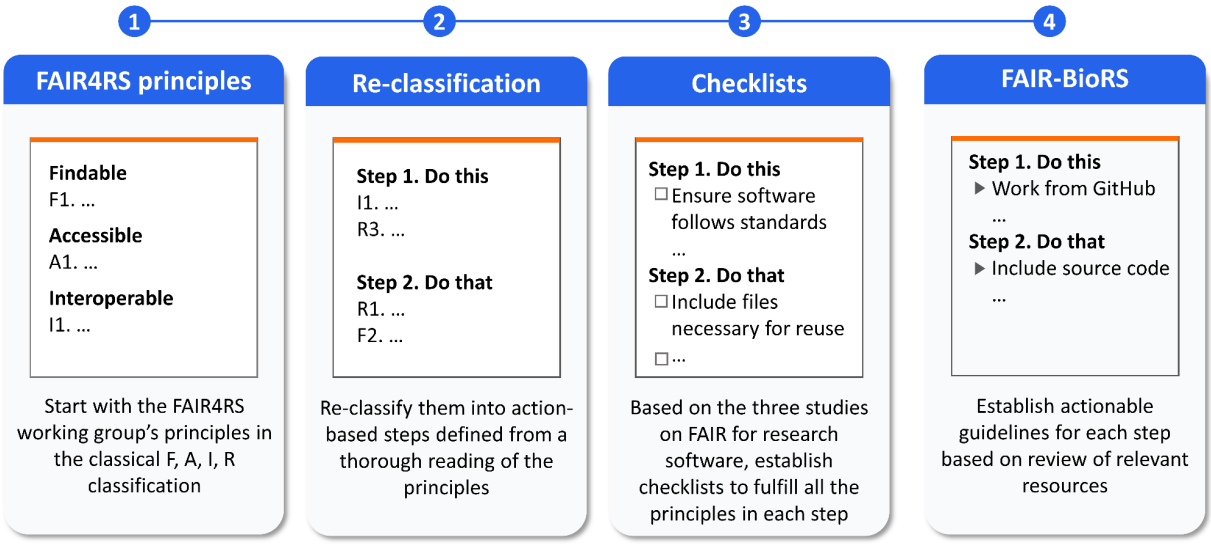

Figure 2. Overall process followed to establish the current version of the FAIR biomedical research software (FAIR-BioRS) guidelines.

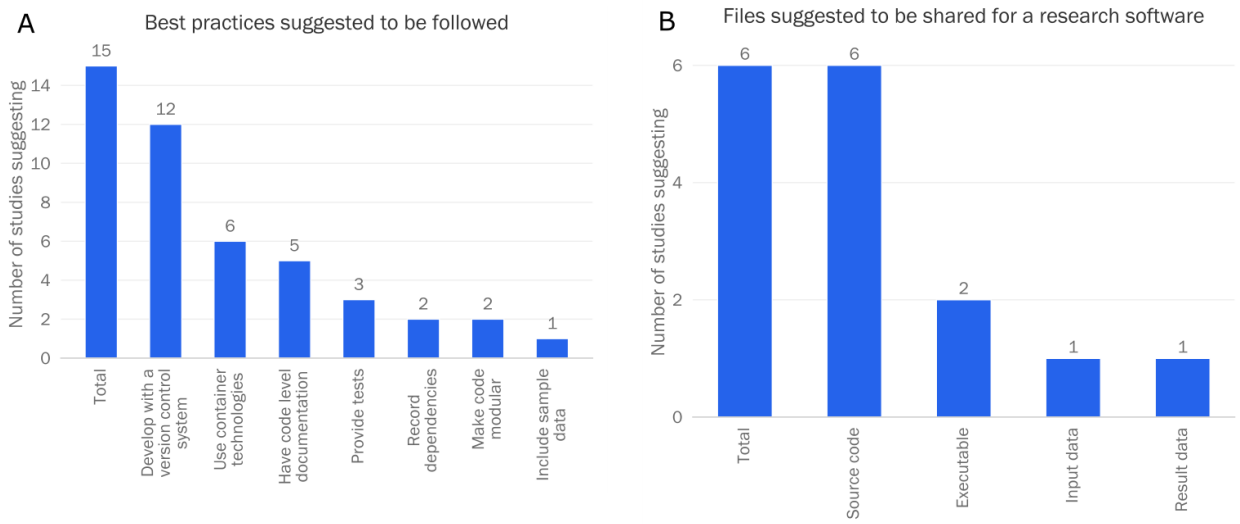

Figure 3. Results from our review showing the best practices suggested to be followed when developing research software (A) and the files suggested to be included when sharing a research software (B).

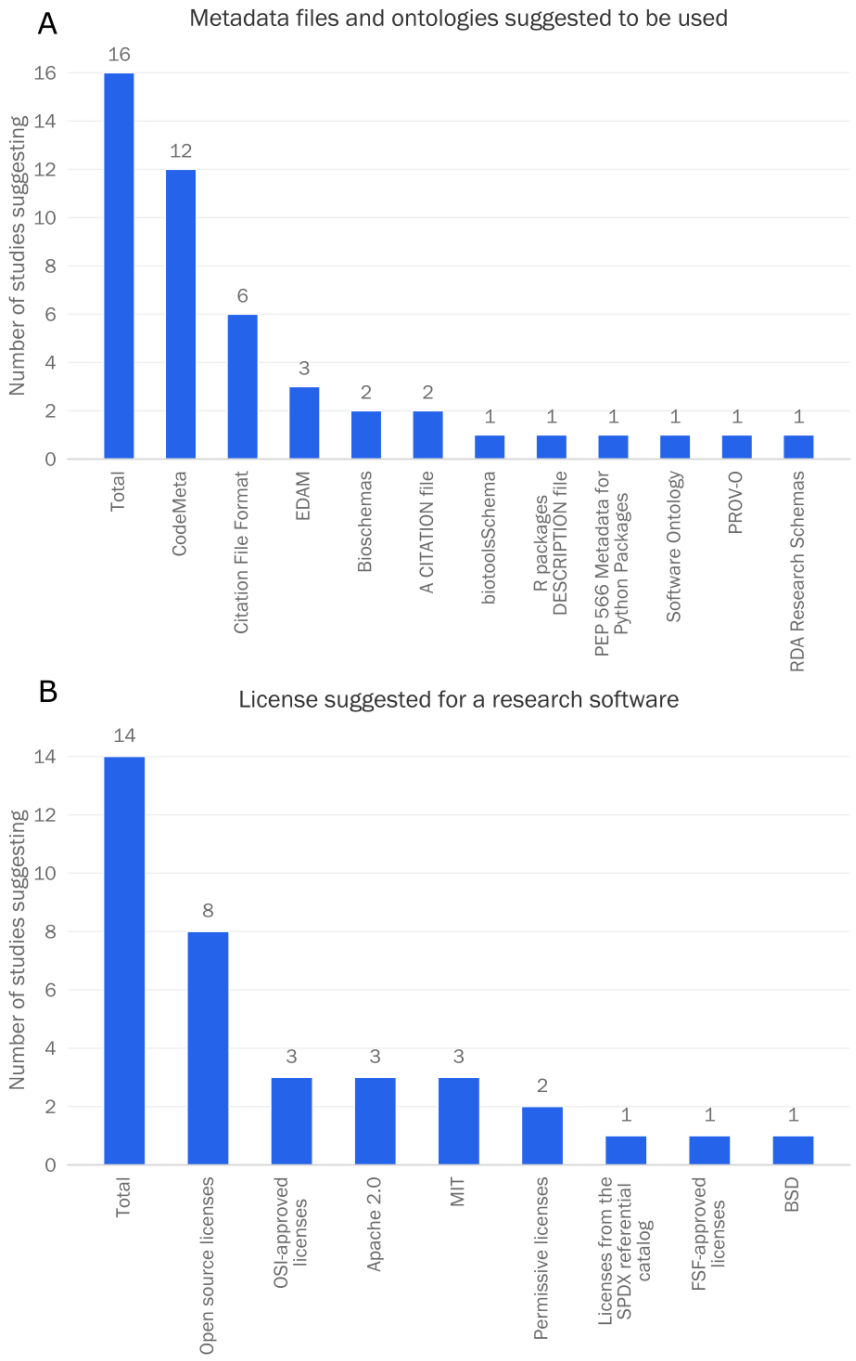

Figure 4. Results from our review showing the metadata files and ontologies suggested (A) and the licenses suggested (B) when sharing a research software.

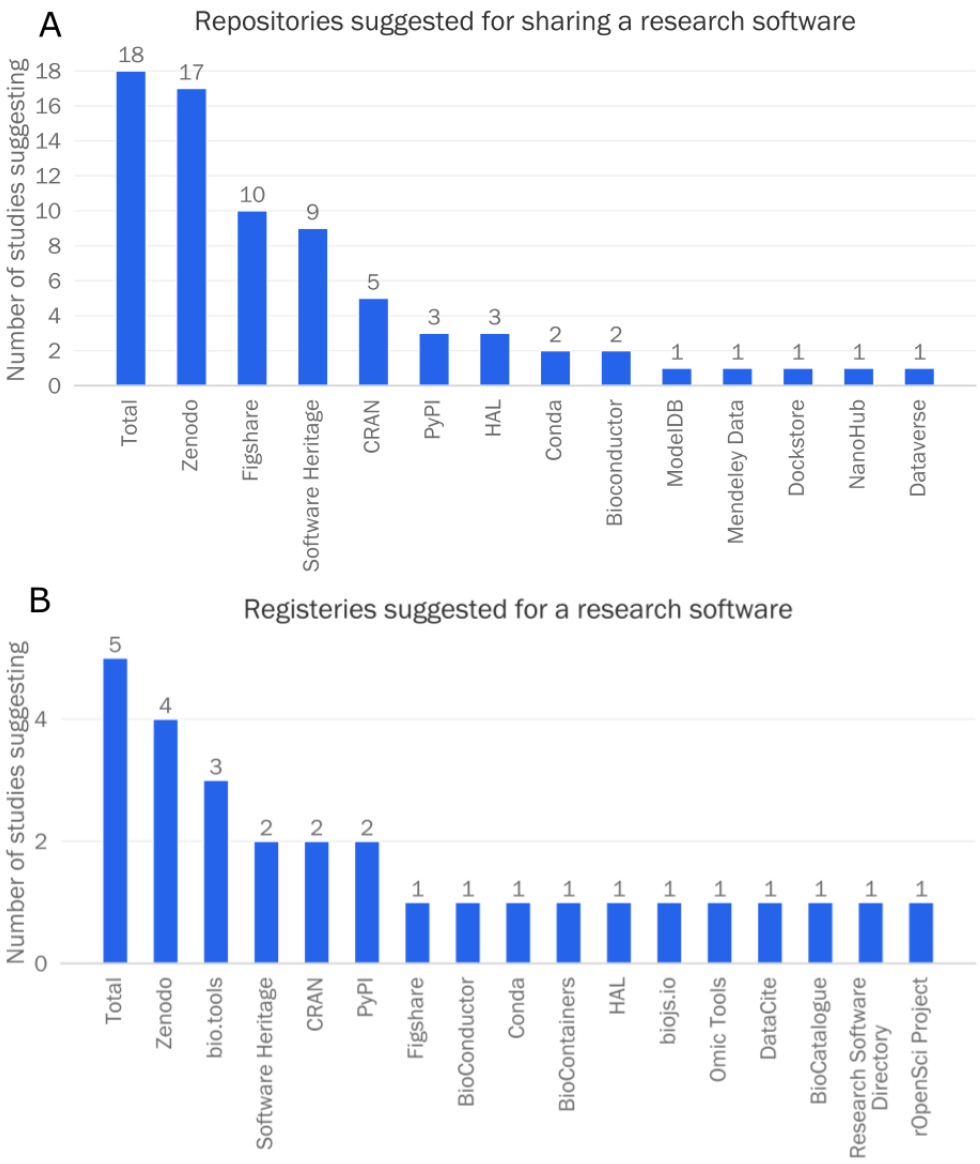

Figure 5. Results from our review showing the repositories suggested (A) and the registries suggested (B) for a research software.

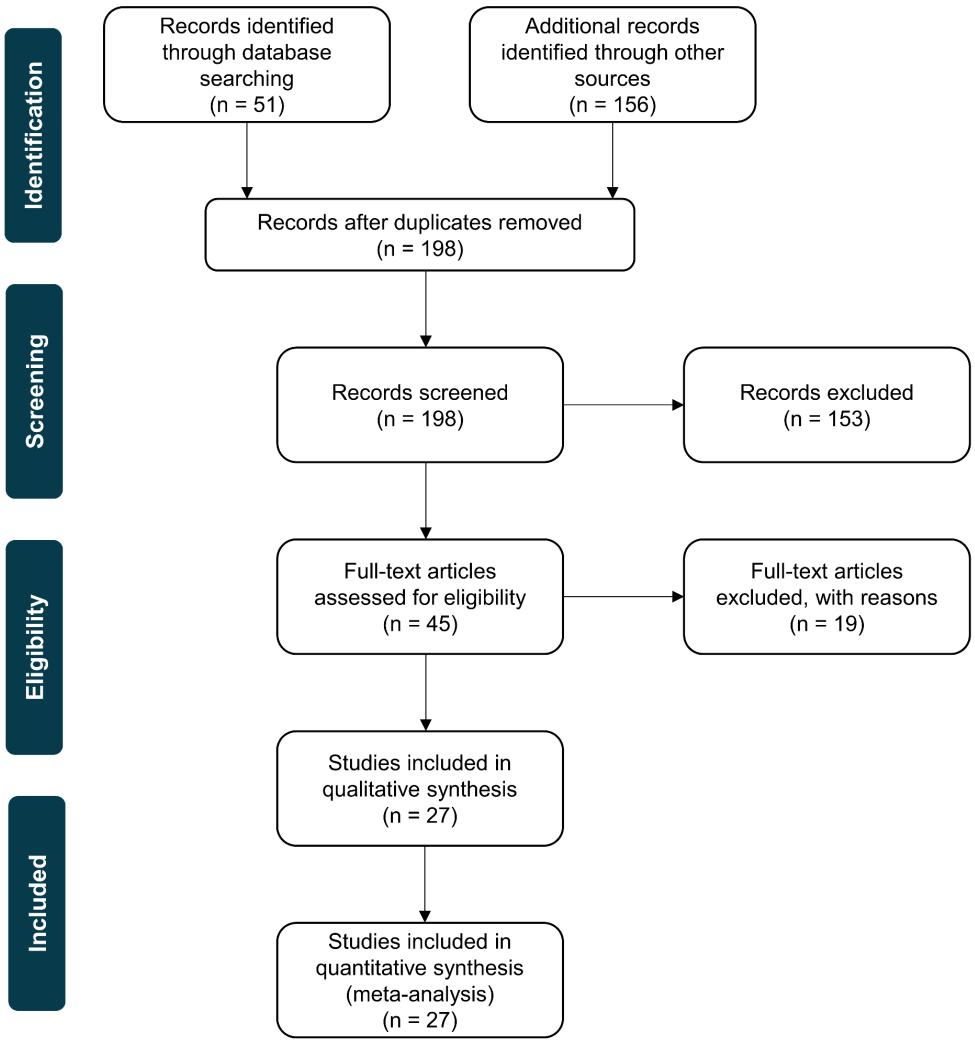

Figure 6. PRISMA diagram providing an overview of our literature review process.

Figure 7. Illustration of the step-by-step guided process implemented in FAIRshare for making biomedical research software FAIR.

Table 1. Classification of the 17 FAIR4RS guiding principles under actionable steps for making research software FAIR, along with actionable checklists to comply with them.

FAIR4RS principles	Actionable checklists
Step 1: Follow standards and best practices during development of the software	
I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards. R3. Software meets domain-relevant community standards	☐ Develop the software such that it reads, writes, and exchanges data in applicable standard formats that are described using controlled vocabulary ☐ Ensure that the software, including its documentation and license, meets domain-relevant community standards and coding practices (for example choice of programming language, standards for testing, usage of file formats, etc). Community standards may include ways of managing and structuring the code, and expectations around the presence and structure of the documentation.
Step 2: Collect files to share	
N/A	☐ Share all necessary files for reuse of the software
Step 3: Include metadata files	
F2. Software is described with rich metadata F3. Metadata clearly and explicitly include the identifier of the software they describe I2. Software includes qualified references to other objects R1. Software is described with a plurality of accurate and relevant attributes R1.2. Software is associated with detailed provenance R2. Software includes qualified references to other software	☐ Provide descriptive metadata following applicable community standards and using controlled vocabularies ☐ Where possible, provide multiple terms for the same, similar, or overlapping concepts • Include at least the following items in the software metadata: ☐ Mention the software's globally unique and persistent identifier and describe how the software can be obtained ☐ Indicate the provenance of the software: Authors, Organization, Release date, Tools and settings used for developing, Standards applied, How to contribute to the software, How to cite it ☐ Include appropriate references to other software/dependencies (requirements, imports, libraries, etc.) which are necessary to compile and run the software using their identifiers ☐ Document parameters, and data required to run the software. This should also include reference to non-digital objects that have a virtual presence in digital systems (e.g., samples, reagents, etc.). Use identifiers and/or controlled vocabularies to do so. Possibly include a resolvable reference to further information about the source. ☐ Document the inputs and outputs of the software
Step 4: Choose a license	
R1.1. Software must have a clear and	☐ Provide a license. The license should be as open as possible,

accessible license	although free and open software (FOSS) is not a requirement of the FAIR principles. ☐ Ensure this license is compatible with the requirements of the licenses of the software's dependencies.
Step 5: Share software on a repository	
F1. Software is assigned a globally unique and persistent identifier F1.1. Different components of the software must be assigned distinct identifiers representing different levels of granularity F1.2. Different versions of the same software must be assigned distinct identifiers F4. Metadata are FAIR and are searchable and indexable A1. Software is retrievable by its identifier using a standardized communications protocol A1.1. The protocol is open, free, and universally implementable A1.2. The protocol allows for an authentication and authorization procedure, where necessary	☐ Associate a unique identifier with each version of the software. This could be achieved by sharing it on a suitable repository. The choice of the repository may be influenced by the programming language used and/or the operating system most used by the respective community. What is considered a different version is up to the owner of the software. ☐ Assign a distinct identifier for different components of the software when possible. ☐ Select a repository that allows access to the software through standard communication protocol such as HTTP or SSH. There should be no restriction on implementing the protocol (open), and no fees to implement it (free). The protocol must handle any authentication and authorization procedure for accessing the software when applicable (e.g., signing a non-disclosure agreement).
Step 6: Register software on a registry	
A2. Metadata are accessible, even when the software is no longer available	☐ Make metadata FAIR and accessible even when the software is not. This could be achieved by storing the metadata in a software repository that stores the metadata separately from the data or in a dedicated metadata registry.

Table 2. The FAIR-BioRS guidelines version 1.0.1. These guidelines provide a step-by-step process along with reference to relevant resources and tools for making biomedical research software FAIR according to the FAIR4RS principles.

Step 1: Follow standards and best practices during development of the software
Step 1.a. Follow general standards and best practices for scientific software, such as working from a version control system (e.g., GitHub, Bitbucket, GitLab), having code level documentation (in code comments, description in the headers), and recording dependencies in a requirement.txt file or a README file. It is recommended to read the “Good enough practices in scientific computing”[22] and the “General guidelines for biomedical software development”[24]. **Step 1.b**. Follow language-specific standards and best practices. They will depend on the development stack used (e.g., the Python Developer’s Guide[41] for Python code, Google’s R Style Guide[42] for R code). **Step 1.c**. Follow standards and best practices for documenting your software. It is suggested to maintain the documentation in a README.md or README.txt file located in the root directory of the software. The following aspects must be documented: inputs and outputs of the software, parameters and data required to run the software, the standards applied, and how to contribute to the software. It is suggested to read the paper on the “Ten simple rules for documenting scientific software”[32].
Step 2: Collect files to share
Include the source code when possible and also include executables if applicable.
Step 3: Include metadata files
Include both codemeta.json[47] and CITATION.cff[131] metadata files in the root directory of the software. The CodeMeta generator[49,50] and CFF file initializer[55,56] are available to help prepare both. Provide, at least, the following fields in the codemeta.json file: Software name (“name”), Software description/abstract (“description”), Unique identifier ("identifier"), Authors (“givenName”, “familyName”) with their Organization name (“affiliation”), Keywords (“keywords”), Programming Language (“programmingLanguage”), First and current release date (“dataPublished” and “dateModified”), License used (“license”). Provide, at least, the following fields in the CITATION.cff file: Authors ("given-names", “family-names") with their Organization name (“affiliation”), Software description/abstract (“abstract”), Unique identifier ("identifiers"), Keywords (“keywords”), License (“license”), Release date (“date-released”).
Step 4: Choose a license
Include the license terms in a LICENSE file located in the root directory of the software. While the FAIR4RS principles do not require research software to be open-source, it is highly recommended using a license approved by the Open Source Initiative (OSI)[132]. Amongst those licenses, it is encouraged to use the permissive MIT[73] or Apache 2.0[74] licenses. Use “Choose a license” for help[71]. Ensure that the selected license is compatible with the requirements of the licenses of the software’s dependencies.
Step 5: Share software on a repository
Step 5.a. If applicable, share software on a language-specific repository (e.g., PyPI[89] or Conda[90] for Python packages, CRAN for R packages[91], Dockstore for Docker-based tools[92]) or a biomedical-specific repository (e.g., ModelDB for computational neuroscience models[93,94] and Bioconductors for R-packages aimed at the analysis of genomics data[95–97]). The Registry of Research Data Repositories can be used to find a suitable repository[75]. **Step 5.b**. Share software on Zenodo[76] or Figshare[83].
Step 6: Register software on a registry

Optionally register the software on bio.tools[98] or on the Research Resource Identifiers (RRID) Portal[99].

Table 3. Assessment of the compliance of the FAIR-BioRS guidelines version 1.0.1 with the FAIR4RS principles.

FAIR4RS principles	Corresponding steps of FAIR-BioRS guidelines
Findable	
F1. Software is assigned a globally unique and persistent identifier	Step 5.b
F1.1. Different components of the software must be assigned distinct identifiers representing different levels of granularity	Step 5.a, Step 5.b. (assuming different levels are shared separately)
F1.2. Different versions of the same software must be assigned distinct identifiers	Step 5.b
F2. Software is described with rich metadata	Step 3, Step 5, Step 6
F3. Metadata clearly and explicitly include the identifier of the software they describe	Step 3, Step 5.b
F4. Metadata are FAIR and are searchable and indexable	Step 3, Step 5, Step 6
Accessible	
A1. Software is retrievable by its identifier using a standardized communications protocol	Step 5
A1.1. The protocol is open, free, and universally implementable	Step 5
A1.2. The protocol allows for an authentication and authorization procedure, where necessary	Step 5
A2. Metadata are accessible, even when the software is no longer available	Step 5.b., Step 6
Interoperable	
I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards.	Step 1.a, Step 1.b
I2. Software includes qualified references to other objects	Step 1.a, Step 1.b
Reusable	
R1. Software is described with a plurality of accurate and relevant attributes	Step 1.c., Step 3, Step 5, Step 6
R1.1. Software must have a clear and accessible license	Step 4
R1.2. Software is associated with detailed provenance	Step 1.c., Step 3, Step 5, Step 6
R2. Software includes qualified references to other software	Step 1, Step 3
R3. Software meets domain-relevant community standards	Step 1