

Project Title	Expanding FAIR solutions across EOSC
Project Acronym	FAIR-IMPACT
Grant Agreement No.	101057344
Start Date of Project	2022-06-01
Duration of Project	36 months
Project Website	fair-impact.eu

Comparison of tools for automated FAIR software assessment

Work Package	WP5 - Metrics, certification, and guidelines
Lead Author (Org)	Mario Antonioletti (UEDIN-SSI), Chris Wood (UEDIN-SSI)
Contributing Author(s) (Org)	Neil Chue Hong (UEDIN-SSI), Elena Breitmoser (UEDIN-SSI), Kara Moraw (UEDIN-SSI), Maaïke Verburg (KNAW-DANS)
Date	2024-08-08
Version	V1.0

Dissemination Level

<input checked="" type="checkbox"/>	PU: Public
<input type="checkbox"/>	PP: Restricted to other programme participants (including the Commission)
<input type="checkbox"/>	RE: Restricted to a group specified by the consortium (including the Commission)
<input type="checkbox"/>	CO: Confidential, only for members of the consortium (including the Commission)



Versioning and contribution history

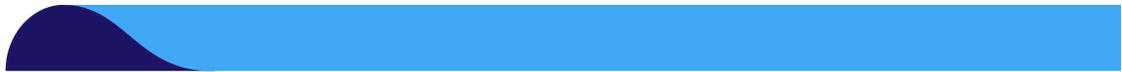
Version	Date	Author	Notes
0.1	05.05.2023	Neil Chue Hong, Elena Breitmoser (UEDIN-SSI)	TOC and V0.1
0.2	01.08.2023	Mario Antonioletti, Chris Wood, Elena Breitmoser, Neil Chue Hong (UEDIN-SSI)	Draft split from D5.2
0.3	30.05.2024	Mario Antonioletti (UEDIN-SSI)	Revised draft for internal task reviewers
0.4	31.07.2024	Elena Breitmoser, Mario Antonioletti, Chris Wood, Kara Moraw, Neil Chue Hong (UEDIN-SSI)	Revised draft for internal FAIR-IMPACT review
1.0	08.08.2024	Elena Breitmoser, Mario Antonioletti, Chris Wood, Kara Moraw, Neil Chue Hong (UEDIN-SSI), Maaïke Verburg (DANS)	Finalised and published on Zenodo

Disclaimer

FAIR-IMPACT has received funding from the European Commission's Horizon Europe funding programme for research and innovation programme under the Grant Agreement no. 101057344. The content of this document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of such content.

Table of Contents

Versioning and contribution history	2
Table of Contents	3
TERMINOLOGY	4
1 Introduction	5
2 Rationale for this report	6
3 FAIR4RS principles and domain subject specificity	9
4 The tools examined	12
5 Evaluation methodology	16
6 Result of the comparison of the tools	17
7 Analysis	23
8 Conclusions and next steps	28
References	29
Appendix 1: howfairis tests	31



TERMINOLOGY

Terminology/Acronym	Description
API	Application Programming Interface
DOI	Digital Object Identifier
FAIR	Findable Accessible Interoperable Reusable
FAIR4RS	FAIR for Research Software
FsF	FAIRsFAIR
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IGSN	International Geo Sample Numbers
PID	Persistent Identifier
RRID	Research Resource Identifier
SCP	Secure CoPy
SFTP	Secure File Transfer Protocol
URL	Uniform Resource Locator

1 Introduction

This document summarises work by the FAIR-IMPACT project to examine the application and potential repurposing of three existing automated assessment tools built to assess FAIR data principles (Wilkinson *et al.*, 2016), to assess compliance with the FAIR for Research Software (FAIR4RS) principles (Chue Hong *et al.*, 2022). At the time this activity was carried out (2023), no automated assessment tools explicitly assessed the FAIR4RS principles, though there are some principles shared by FAIR data and software, in part due to the lack of metrics for FAIR software at the time. The viability of repurposing these tools is based on the similarity between the corresponding FAIR principles applied to data and research software. However, sufficient differences exist that make the direct assessment of FAIR4RS from a FAIR data perspective difficult for some of the principles. In addition, a software assessment tool that uses a different set of recommendations for software based on the FAIR principles was also examined. After the initial explorations presented in this output, FAIR-IMPACT Deliverable 5.2 was created to propose a set of metrics for assessing the FAIR4RS principles (Chue Hong *et al.*, 2023). A companion document (Moraw *et al.*, 2024) extends one of the existing FAIR data assessment tools to provide assessments of research software against the FAIR4RS principles. This exploratory activity has now been published as a companion output to this line of work, presenting enriching background context as well as recommendations for tool developers and users towards a further development of FAIR assessment tools.



2 Rationale for this report

Several tools have been, or are in the process of being, developed to provide an automated way to determine compliance against the FAIR data principles (Wilkinson *et al*, 2016). As the FAIR principles are advisory and not prescriptive, those interested in performing automated assessments of the FAIR principles have for each principle proposed one or more abstract criteria to evaluate whether a FAIR principle has been met. These criteria are referred to as metrics. For example, Figure 1 shows the structure adopted to specify metrics by FAIRsFAIR¹ (FsF). Note that different levels of compliance or maturity indicators (Bahim *et al*, 2020) may also be defined as part of a metric requiring different levels of testing. Others employ similar but slightly different templates for their metrics, see (Wilkinson *et al*, 2018).

Field	Description
Metric Identifier	The local (FAIRsFAIR) identifier of the metric (following a prescribed naming convention).
Metric Name	Metric name in a human-readable form.
Description	The definition of the metric, including examples.
FAIR principle	The FAIR principle most related to the metric.
Assessment	Requirements and methods to perform the assessment against the metric.
Comments	A list of related resources which may be used as a reference basis to implement the assessment, constraints and limitations of the proposed assessment

Figure 1: Structure of an FsF metric (Devaraju *et al*, 2022)

To perform an automated assessment of a principle, each metric has to be turned into one or more specific tests. This is a practical way of checking whether the metric is satisfied. Finally, the tests can be scripted so that they can be used by the corresponding tool to evaluate the level of FAIR compliance. This hierarchy is schematically shown in Figure 2.

As a case in point, we can take the principle “A1.2: *The protocol allows for an authentication and authorisation procedure where necessary*”, identical for FAIR data and FAIR4RS, for which metrics could be developed:

- Is the resource available through HTTP(S)?
 - An attempt to access a URL/DOI/etc with HTTP(s) returns a 200, 202, 203 or 206 HTTP response.
- Is the code available through other authenticated protocols, such as ftp, sftp, scp, etc?

¹ FAIRsFAIR <https://www.fairsfair.eu> last accessed 01/04/24.

In practice, testing for the availability of HTTP(S) may suffice as most software repositories support HTTP(S) and the protocol supports the authentication and authorisation requirements. The tests need to be specific to be applicable and may result in introducing normative behaviour to effect compliance, e.g. look for a LICENSE file, i.e. license is both a verb and a noun in US English but other forms of English use licence for the noun, but may not accept LICENCE or LICENSE.txt as satisfying this test. It is also important to note that the communities generating and using these metrics need to observe a level of coherency and alignment for the assessment of each FAIR principle so that the different tools provide consistent results, see Wilkinson *et al* (2022).

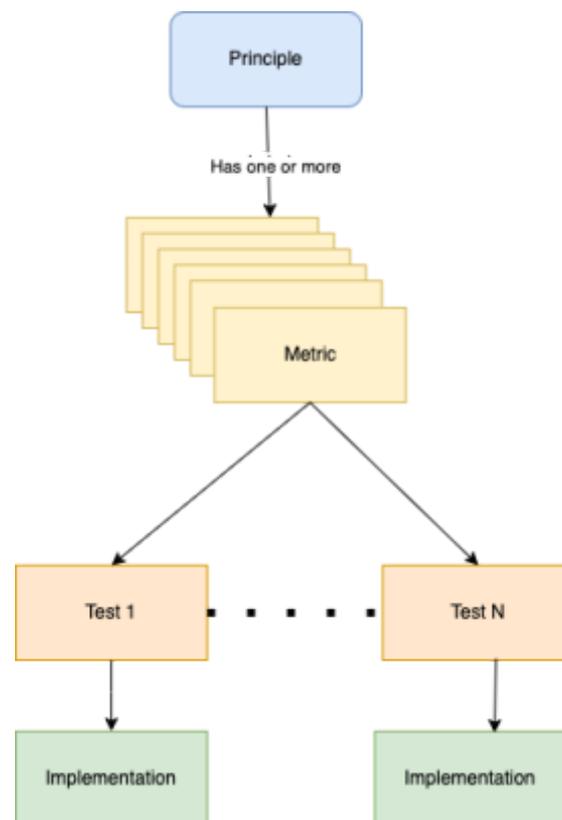


Figure 2: Principles to metrics, to tests and implementations.

Other FAIR principles are being developed for other specific types of digital objects, such as computational workflows², ontologies³, etc. In particular, groups involved in developing automated assessment tools for the FAIR4RS principles (Chue Hong *et al*, 2022) are in the process of establishing metrics to be able to assess the FAIRness of Research Software and incorporate these into the same tooling that has been developed to assess FAIR data. The motivation for this part of the work was thus partly driven by a need to choose a tool that

² See <https://workflows.community/groups/fair/> last accessed 29/05/24.

³ See <https://www.excelra.com/blogs/ontologies-and-the-fair-data-principles/> last accessed 29/05/24.

would facilitate an attempt to incorporate the FAIR4RS principles and inform the provision of a set of metrics for FAIR4RS tailored to a specific subject domain - the social sciences (Chue Hong *et al*, 2023) and suggest a tool that could be extended to implement these metrics as tests (Moraw *et al*, 2024).

In contrast, an alternative approach taken for a FAIR compliance assessment is to let users perform self-assessments based on guided questionnaires of their repositories to evaluate how FAIR they are for data such as in the SATISFYD⁴ questionnaire or to use the “Self-assessment for FAIR research software” questionnaire⁵ for FAIR4RS. Through a limited set of questions, a user can become more aware of the issues involved in increasing the FAIRness of their digital assets. These variants are noted in passing here and are not considered further as the focus here is on tools that can automate the assessment.

Most metrics required to assess the FAIR principles, both for data or research software, are largely domain-agnostic so specific domain knowledge is not required. In the next section, we briefly consider metrics requiring domain knowledge to properly assess those particular principles.

⁴ See <https://satisfyd.dans.knaw.nl> last accessed 31/01/24.

⁵ See <https://fairsoftwarechecklist.net/v0.2/> last accessed 31/01/24.





3 FAIR4RS principles and domain subject specificity

Most metrics do not require specific domain knowledge. For instance, from the FAIR4RS principles, F1 states that: *“software is assigned a globally unique and persistent identifier”* and A1.1 states that *“the protocol is open, free, and universally implementable”* could both be tested in a general way although there is always the possibility that domain-specific communities could add their domain-specific identifiers as a requirement, e.g. IGSN (International Geo Sample Numbers) for geoscientists, RRDs (Research Resource Identifiers) for biologists, etc. see (Plomp, 2020). On the other hand, some of the FAIR4RS principles do necessitate input from the domain being tested if anything other than a superficial evaluation is to be performed. For instance, I1 states: *“software reads, writes and exchanges data in a way that meets domain-relevant community standards”* and for R3 *“software meets domain-relevant community standards”*. The metrics for these principles would require input from the particular subject domains to be evaluated. Table 1 outlines the FAIR4RS principles that may require domain knowledge input to produce meaningful metrics to assess compliance with the FAIR4RS principles.

Table 1 – Whether domain knowledge may be required to assess the FAIR4RS principles

FAIR4RS principle	Domain agnostic	Requires domain knowledge	Notes
F1: Software is assigned a globally unique and persistent identifier.	✓		Some domains may create domain-specific PIDs.
F1.1: Components of the software representing levels of granularity are assigned distinct identifiers.	✓		
F1.2: Different versions of the software are assigned distinct identifiers.	✓		
F2: Software is described with rich metadata.	✓	✓	Metadata could have domain-specific parts.

F3: Metadata clearly and explicitly include the identifier of the software they describe.	✓		There might be domain-specific identifiers.
F4: Metadata are FAIR, searchable and indexable.	✓		
A1: Software and its associated metadata are accessible by their 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.	✓		
A2: Metadata are accessible, even when the software is no longer available.	✓		One may have to search domain-specific registries.
I1: Software reads, writes and exchanges data in a way that meets domain-relevant community standards.		✓	
I2: Software includes qualified references to other objects. ⁶	✓		There may be domain-specific identifiers.
R1: Software is described with a plurality of accurate and relevant attributes. ⁷	✓	✓	Attributes may be domain-specific.
R1.1: Software is given a clear and accessible license.	✓		
R1.2: Software is associated with detailed provenance.	✓		
R2: Software includes qualified	✓		

⁶ In the FAIR4RS principles, I2 is derived from but slightly different from FAIR principle I3 - the number change may be confusing.

⁷ Testing for the accuracy of the metadata is harder than testing for the existence of it.

references to other software.			
R3: Software meets domain-relevant community standards.		✓	

At the time the work reported in Section 4 was carried out (2023), tools and metrics used to assess the FAIR4RS principles were at a very early stage of development. Of the four tools examined, three were developed to provide FAIR data assessments while the remaining tool examined the FAIRness of software but did not use the FAIR4RS principles, relying instead on five generic principles that can be mapped to the FAIR4RS principles. The focus of this evaluation was to determine whether the existing tools and/or metrics could be used out of the box to test compliance with the FAIR4RS principles and whether they might be extended to specifically include the FAIR4RS principles in a domain-specific context as specified in (Chue Hong *et al*, 2023). The next section describes the four tools examined in more detail.

4 The tools examined

The four FAIR assessment tools examined are listed in Table 2. Three of these tools were originally developed to test compliance with the FAIR data principles, FAIR-Enough (Emonet and Dumontier), F-UJI (Devaraju and Huber, 2021) and FAIR-Checker⁸ (Gaignard *et al*, 2023). The fourth tool, howfairis (Spaaks *et al*, 2022), targets software but does not explicitly use the FAIR4RS principles in its assessment; instead, it relies on five recommendations for FAIR software developed by the Netherlands eScience Center⁹ and the Data Archiving and Networked Services¹⁰ (DANS), a Dutch centre that advises on the sustainable storage and sharing of data. The source code for all of these tools are available on GitHub.

Table 2 – FAIR assessment tools evaluated

Tool	Version	Licence	Implementation	Targets	Date of release ¹¹
FAIR-Enough ¹²	API 0.1.0	MIT	Python/ Typescript ¹³	data	No releases ¹⁴
F-UJI ¹⁵	2.0.2	MIT	Python	data	07/10/22
FAIR-Checker ¹⁶	1.2.3	MIT	Python	data	01/06/23
howfairis ¹⁷	0.14.2	Apache 2.0	Python	software	01/09/22

where:

- **Tool:** the name of the tool.
- **Version:** the version of the tool examined.
- **Licence:** the distribution licence for the tool.
- **Implementation:** the implementation language used.
- **Targets:** whether the tool assesses data or software.
- **Date of release:** the release date of the version examined.

⁸ FAIR-Checker is funded by the French Institute for Bioinformatics (IFB) through the [PIA2 11-INBS-0013 grant](#).

⁹ <https://www.esciencecenter.nl> last accessed on 30/11/23.

¹⁰ <https://dans.knaw.nl> last accessed on 30/11/23.

¹¹ Using the date format: dd/mm/yy.

¹² FAIR-Enough <https://github.com/vemonet/fair-enough-metrics> last accessed 17/08/23.

¹³ Python is used for the back-end <https://github.com/vemonet/fair-enough-metrics> and functionality and Typescript for the web interface for the tool.

¹⁴ FAIR-Enough haven't used the "Releases" feature of Github to release a numbered version of their tool. Although the API, which is published on a third-party site, does have a version associated with it (0.1.0) it is unclear whether this uses the current version of the code committed to GitHub

¹⁵ F-UJI <https://github.com/pangaea-data-publisher/fuji> last accessed 17/08/23.

¹⁶ FAIR-Checker <https://github.com/IFB-ElixirFr/FAIR-checker> last accessed 17/08/23.

¹⁷ Howfairis <https://github.com/fair-software/howfairis> last accessed 17/08/23.

The evaluations were carried out between April and July 2023.

Table 3 describes the metrics used by each tool to make the FAIRness assessment - see Mons *et al* (2017) for a discussion about what FAIR compliance means. The tools provide some guidance on how a user might improve their level of FAIRness.

Table 3 - metrics used by the assessment tools

Tool	Metrics used
FAIR-Enough	<p>Can choose four types of metrics¹⁸ in their web interface for their tool conforming to the specifications defined by the FAIRMetrics working group¹⁹:</p> <ul style="list-style-type: none"> ● FAIR Enough metadata maturity indicators (fair-enough-metadata) ● FAIR Enough maturity indicators (fair-evaluator-maturity-indicators) ● FAIR Enough data maturity indicators (fair-enough-data) ● FAIR Enough maturity indicators for Rare Diseases (rare-disease-maturity-indicators) <p>FAIR-Enough also encourages users to add their tests to the FAIR-Enough testing framework.</p>
F-UJI	Uses the FAIRsFAIR Data Object Assessment Metrics (Devaraju <i>et al</i> 2022).
FAIR-Checker	Uses the Go-FAIR metrics ²⁰ .
howfairis	<p>Uses a set of criteria developed by the Netherlands eScience Center and DANS for the following five recommendations:</p> <ol style="list-style-type: none"> 1. Use a publicly accessible repository with version control 2. Include a License 3. Register your code in a community registry 4. Enable citation of the software 5. Use a software checklist

Table 4 lists the criteria²¹ used by the howfairis FAIR software recommendations and shows how we mapped these to the FAIR4RS principles.

¹⁸ For a description of each of the collection of metrics sets, authors and dates for the metric sets below see <https://fair-enough.semanticscience.org/collections> last accessed 26/03/68.

¹⁹ See <https://github.com/FAIRMetrics/Metrics> for a YAML description targeting a smartAPI interface description of the metrics and <https://fairsharing.github.io/FAIR-Evaluator-FrontEnd/#!/metrics> for a description of the metrics, links last accessed 06/09/23.

²⁰ Go FAIR metrics <https://www.go-fair.org/fair-principles> last accessed 04/09/23.

²¹ The criteria used to test the recommendations are given in Appendix 1.

Table 4 - mapping the Netherlands eScience Center and DANS recommendations to the FAIR4RS principles

Recommendation	FAIR4RS principles
Use a publicly accessible repository with version control	A1 (software and its associated metadata are accessible by their identifier using a standardized communications protocol) ²² , A1.1 (the protocol is open, free, and universally implementable) ²³ , A1.2 (the protocol allows for an authentication and authorization procedure, where necessary) ²⁴ , R1.2 (software is given a clear and accessible license) ²⁵
Add a License	R1.1 (software is given a clear and accessible license)
Register your code in a community registry	R1.2 (software is given a clear and accessible license) ²⁶
Enable citation of the software	R1.2 (software is given a clear and accessible license)
Use a software checklist (presence of the OSSF badge ²⁷)	F1 (software is assigned a globally unique and persistent identifier), F1.2 (different versions of the software are assigned distinct identifiers), F2 (software is described with rich metadata), F3 (metadata clearly and explicitly include the identifier of the software they describe), F4 (metadata are FAIR, searchable and indexable), and R3 (software meets domain-relevant community standards)

We have made some assumptions to compile Table 4, for instance, the scope and interpretation of what finding a specific badge type in a “README.md” file means and how it maps to one or more of the FAIR4RS principles.

From our interpretation of how the Netherlands eScience Center and DANS recommendations map to the FAIR4RS principles we note that howfairis does not check for:

²² Limited to canonical GitHub and GitLab instances. It does not check whether any sourcecode is present

²³ The tool only supports GitHub and GitLab repositories which by extension only supports https and ssl.

²⁴ This is limited to canonical GitHub and GitLab only. Both of these support https and private repositories - the protocol and infrastructure support authentication and authorisation, API keys for both frameworks through the environment variables, where a user and access token can be included. This would be an implicit check.

²⁵ Implicitly - there will be a commit log if hosted in a git repository which howfair does not explicitly check.

²⁶ Checks are made, though not exhaustive, that software has been contributed to software community distribution points, e.g. CRAN for R packages, PyPI for Python, etc.

²⁷ Criteria for getting the best practice badge: <https://bestpractices.coreinfrastructure.org/en/criteria/0>. Last accessed 14/07/23.

- F1.1 (“components of the software representing levels of granularity are assigned distinct identifiers”),
- F4 (“Metadata are FAIR, searchable and indexable”),
- I1 (“software reads, writes and exchanges data in a way that meets domain-relevant community standards”) and
- I2 (“Software includes qualified references to other objects”).

In the next section, we discuss how the tools were evaluated.

5 Evaluation methodology

Each tool was initially downloaded, installed and run on a local machine. However, at the time of writing all the FAIR data assessment tools also provided a public web service access point, see Table 5, through which an evaluation of a repository could be performed by providing a URL to the resource to be evaluated. Thus, instead of using local deployments, the services provided by the tool developers were used instead. Only howfairis needed to be run on a local machine to perform its evaluation.

Table 5 - web endpoints to do FAIR data assessments for the tools used

Tool	Service
FAIR-Enough	https://fair-enough.semanticscience.org/
F-UJI	https://www.f-uji.net/
FAIR Checker	https://fair-checker.france-bioinformatique.fr/check

The tools were run against several end-points:

- A public software repository²⁸ was set up and deployed on GitHub, GitLab and in a local institutional deployment of GitLab. The aim was to use the tool feedback to improve the FAIRness of the repository but this objective was quickly discarded after it proved to be difficult to translate the feedback into actionable steps. Instead, the output from each tool was examined, as well as the metrics that the tool used and whether these could be extended to meet the FAIR4RS principles.
- Each tool was run against the repositories that hosted it.
- F-UJI was run against a sample dataset²⁹ as well as against this dataset’s metadata and assessed results.

In practice, the output produced by each tool was used for the evaluation discussed in the next section.

²⁸ The test repository is deployed at <https://github.com/marioa/fair-test>, <https://gitlab.com/marioa/fair-test> and <https://git.ecdf.ed.ac.uk/mario/fair-test> last accessed on 27/10/23.

²⁹ Sample dataset <https://doi.org/10.5285/16f52064-a19d-4cf5-a388-aff04a592179> last accessed 30/11/23.

6 Result of the comparison of the tools

Table 6 outlines what the metrics used by each tool test for and whether they might also be deemed to satisfy the corresponding FAIR4RS principle. Most FAIR4RS principles more or less match the corresponding FAIR data principle numbering except that the I3 FAIR data principle maps to the I2 FAIR4RS principle.

Table 6 - the result of the comparison for each FAIR4RS and how the metrics applied by the tooling might be interpreted in a FAIR4RS assessment

Principle	Tools Evaluation method summary
F1: Software is assigned a globally unique and persistent identifier.	<p>All four tools check for this principle.</p> <ul style="list-style-type: none"> ● FAIR-Checker checks that it has a reachable URL and will check if the URL is persistent (WebID³⁰, PURL³¹ or a DOI³²). ● FAIR-Enough will do the same but also uses identifiers.org which checks for compact identifiers used in the life sciences. ● F-UJI checks for URL or IRI formats and if these are not resolvable it will check for UUID or hash type syntax. ● howfairis does not explicitly check for the principle but it expects a valid URL to be provided to a software repository hosted in GitHub or GitLab so the identifier check is implicit. It does not check for persistent identifiers.
F1.1: Components of the software representing levels of granularity are assigned distinct identifiers.	<p>None of the examined tools tests this principle as the FAIR4RS guidelines do not have a direct equivalent in the FAIR for data so we would not expect the FAIR for data tools to test this. We did not think any of the criteria used by howfairis mapped to this principle.</p>
F1.2: Different versions of the software are assigned distinct identifiers.	<p>None of the FAIR data tools tests this FAIR4RS principle. Compliance for howfairis is assumed through the presence of the OSFF badge in the software’s README.md file (although the absence of the OSFF badge does not mean that F1.2 for the FAIR4RS has not been met). This applies to all the tests performed by howfairis, while the presence of a badge might indicate that the principle is met the inverse is not true, that is, if the badge is not there then the principle has not been met is not true.</p>
F2: Software is	<p>We implicitly infer compliance with this FAIR4RS principle for</p>

³⁰ See <https://www.w3.org/2005/Incubator/webid/spec> last accessed on 28/03/24.

³¹ See https://en.wikipedia.org/wiki/Persistent_uniform_resource_locator last accessed 28/03/24.

³² See <https://doi.org/10.1000/182> last accessed 28/03/24.

<p>described with rich metadata.</p>	<p>howfairis through the presence of the OSSF badge which acts as a proxy for a number of the FAIR4RS principles.</p> <p>The other three tools examined focus on the FAIR principles for data but could be modified to take into account the requirements for FAIR4RS.</p> <ul style="list-style-type: none"> ● FAIR-Checker checks for the existence in the metadata of at least one of <i>Dublin Core</i>³³ (dct) or <i>Data Catalogue</i>³⁴ (dcat) properties from the list of dct:title, dct:description, dct:accessURL, dct:downloadURL, dcat:endpointURL or dcat:endpointDescription. It also checks that at least one (weak) or all (strong) of the ontology classes or properties used are known in major ontology registries. ● F-UJI checks that the metadata can be obtained via common web methods, that core data citation metadata is available and/or that core descriptive metadata is available but it does not check for the FAIRness of the metadata or whether the metadata uses controlled vocabularies. ● FAIR-Enough tries to find 'grounded' metadata, i.e. that the metadata terms are in a resolvable namespace, where the resolution leads to a definition of the meaning of the term and it also tries to find structured metadata which could, for example, be RDFa, embedded json, json-ld, or content-negotiated structured metadata such as RDF Turtle.
<p>F3: Metadata clearly and explicitly include the identifier of the software they describe.</p>	<ul style="list-style-type: none"> ● FAIR-Enough checks whether the metadata contains a unique identifier to the metadata itself and whether the metadata document contains a globally unique and persistent identifier for the digital resource. It parses the metadata to search for a given digital resource, GUID, and if found, retrieves information about this resource (title, description, date created, etc). It should not be too difficult to extend this to deal with an identifier to the software being described. ● F-UJI checks that the metadata is given in a way that major search engines can ingest it into their catalogues (JSON-LD, Dublin Core, RDFa) and metadata is registered in major research data registries (e.g. DataCite, etc). Again, software-specific extensions should be possible.
<p>F4: Metadata are FAIR, searchable</p>	<ul style="list-style-type: none"> ● FAIR-Enough extracts the title from RDF metadata, or from the DataCite API (in the case it is a DOI), then searches for

³³ See <https://www.dublincore.org/specifications/dublin-core> last accessed 28/03/24.

³⁴ See <https://www.w3.org/TR/vocab-dcat-3> last accessed 28/03/24.



<p>and indexable.</p>	<p>the resource URL in popular search engines using the extracted title: - DuckDuckGo search engine (no limitations, but it misses some scientific data repositories) - Google custom search API (results are not as good as the regular Google search though, limited to 100 queries per day) - Bing search engine (qualitative results, but limited to 1000 queries per months, used as last recourse).</p> <ul style="list-style-type: none"> ● F-UJI checks that the metadata is presented in a way that it can be ingested by search engines and that it can also be found by a major search engine. <p>Neither tool appears to explicitly test that the metadata is FAIR.</p>
<p>A1: Software and its associated metadata are accessible by their identifier using a standardized communications protocol.</p>	<ul style="list-style-type: none"> ● Howfairis only checks if a repository is there, i.e. not a 404, using an HTTPS connection - it does not explicitly check for the associated software metadata. ● F-UJI checks that the landing page link is based on standard web communication protocols (e.g. HTTP(S), sftp, ssh, etc.), information about access restrictions or rights can be identified in metadata, the data access information is machine-readable and the information is indicated by (not machine-readable) standard terms.
<p>A1.1: The protocol is open, free, and universally implementable.</p>	<p>All tools check for this. howfairis does an implicit check, whereas all the other tools check that HTTP(S) or that a URL is being used.</p>
<p>A1.2: The protocol allows for an authentication and authorization procedure, where necessary.</p>	<p>This check could be done by checking that HTTP(S) is used - which FAIR-Enough and FAIR-Checker do.</p> <ul style="list-style-type: none"> ● FAIR-Enough also checks for the <i>Dublin Core</i> (dc) dc:accessRights metadata property, which may point to a document describing the data access process. ● FAIR-Checker checks if the protocol supports authentication and authorisation as well as checking the metadata to see if access rights are specified in metadata through terms odrl:hasPolicy (<i>open digital rights language</i>³⁵), dc:rights, dc:accessRights, or dc:license. ● Howfairis only checks GitHub or GitLab URLs which only use https. howfairis can also check private repositories by putting access credentials (an access token and the corresponding username) in an environment variable. ● F-UJI does not check this principle.

³⁵ See <https://www.w3.org/TR/odrl-vocab> last accessed 28/03/24.

<p>A2: Metadata are accessible, even when the software is no longer available.</p>	<p>This appears to be a hard principle to check. Only FAIR-Enough tries and even then indirectly by testing if the metadata contains a persistence policy, explicitly identified by a persistencePolicy key (in hashed data) or a http://www.w3.org/2000/10/swap/pim/doc#persistencePolicy predicate in Linked Data.</p> <ul style="list-style-type: none"> ● F-UJI checks that the persistent identifier system guarantees the preservation of associated metadata. Currently, only DOIs fulfil that requirement.
<p>I1: Software reads, writes and exchanges data in a way that meets domain-relevant community standards.</p>	<p>No tool checks for this principle for research software.</p>
<p>I2: Software includes qualified references to other objects.</p>	<p>No tool explicitly checks this principle for research software but I3, “(Meta)data include qualified references to other (meta)data”, for the FAIR data principles could be thought of as a slightly generalised version of I2 in FAIR4RS if the (Meta)data is restricted to research software. Currently, the tooling checks that:</p> <ul style="list-style-type: none"> ● FAIR-Enough: the metadata contains outward references³⁶. It uses a maturity indicator to test if the metadata links outward to third-party resources. It only tests metadata that can be represented as Linked Data. It will succeed if there is at least 1 object in the metadata that uses a different host other than the subject URI being evaluated. ● FAIR-Checker: verifies that at least 3 different URL authorities are used in the URIs of RDF metadata. ● F-UJI: checks that <ul style="list-style-type: none"> ○ Metadata includes links between the data and its related entities ○ Related resources are explicitly mentioned in metadata ○ Related resources are indicated by machine-readable links or identifiers <p>These criteria would be specialised to deal with research software.</p>
<p>R1: Software is described with a</p>	<ul style="list-style-type: none"> ● For howfairis this may be inferred by the presence of the OSSF badge and also by the inclusion of a registry badge,

³⁶ See <https://w3id.org/fair-enough/metrics/tests/i3-metadata-contains-outward-links> (Version: 0.1.0) - last accessed 10/01/24.

<p>plurality of accurate and relevant attributes.</p>	<p>i.e. the code has been submitted to a code registry (e.g. PyPI for Python, Cran for R, npm for JavaScript, etc.), from which we infer associated metadata will be included as well as pointers to other software but this is not explicitly tested for.</p> <ul style="list-style-type: none"> ● F-UJI does some pertinent checks in the metadata but none of these are software-related.
<p>R1.1: Software is given a clear and accessible license.</p>	<p>Every tool checks for the existence of a LICENSE file.</p> <p>FAIR-Checker, F-UJI and FAIR-Enough also check for information about the license in the metadata and F-UJI also checks that it is registered at SPDX³⁷.</p>
<p>R1.2: Software is associated with detailed provenance.</p>	<ul style="list-style-type: none"> ● FAIR-Checker verifies that at least one provenance property from PROV, DCTerms, or PAV ontologies is found in the metadata. ● Similarly, F-UJI checks that metadata contains elements which hold provenance information and can be mapped to PROV and that metadata contains provenance information using formal provenance ontologies (PROV-O).
<p>R2: Software includes qualified references to other software.</p>	<p>This is an explicit FAIR4RS requirement and as such none of the tools that test for FAIR data check this principle.</p> <p>Howfairis implicitly does this by checking if the code has been submitted to a software registry such as PyPi, CRAN, etc. If so, dependencies are usually noted. For instance, the DESCRIPTION file in CRAN, Pypi may use a requirements.txt file or a setup.py and similarly for the other package registry. These may/may not satisfy R2. There is no explicit check for the qualified references to other software in howfairis.</p>
<p>R3: Software meets domain-relevant community standards.</p>	<p>We infer compliance for howfairis through the presence of the OSSF badge.</p> <p>For the other tools, the FAIR data principle - R1.3 (Meta)data meet domain-relevant community standards, which could be repurposed to comply with R3.</p> <ul style="list-style-type: none"> ● FAIR-Checker verifies that at least one used ontology class or property (weak version) or that all ontologies and properties (strong version) are known in major ontology

³⁷ Software Package Data eXchange (SPDX) lincense list <https://spdx.org/licenses> last accessed 24/07/23.

	<p>registries (OLS³⁸, BioPortal³⁹, LOV⁴⁰).</p> <ul style="list-style-type: none"> ● F-UJI checks that a community-specific metadata standard is detected using namespaces or schemas found in provided metadata or metadata services outputs, community-specific metadata standard is listed in the re3data⁴¹ record of the responsible repository, multidisciplinary but community-endorsed metadata (RDA Metadata Standards Catalog⁴²) standard is listed in the re3data record or detected by namespace and that the format of a data file given in the metadata is listed in the long term file formats, open file formats or scientific file formats controlled list. <p>So, although R1.3 is not a FAIR4RS principle, it can be repurposed to meet R3 - F-UJI does this and FAIR-Checker partially meets the requirement as its checks are more restrictive.</p> <ul style="list-style-type: none"> ● FAIR-Enough does not explicitly check for R1.3 other than in using the very specific rare-disease-maturity-indicators which apply to rare diseases specifically.
--	---

As we can see from the table the similarities between the FAIR for data principles and the FAIR4RS principles mean that the tooling and/or the metrics could be extended to enable an assessment of research software, see (Moraw *et al*, 2024) for a particular example.

In the next section, we summarise the results to describe whether we think the existing metrics used by the tools already comply with the FAIR4RS, whether they partially comply with the principle with some additional work or whether they do not comply at all as things stand and more work is required in the metrics and/or the tools to be able to assess the corresponding FAIR4RS metric.

³⁸ Ontology look up service, <https://www.ebi.ac.uk/ols4> last accessed 28/03/24.

³⁹ See <https://bioportal.bioontology.org> last accessed 28/03/24.

⁴⁰ Linked Open Vocabularies, <https://lov.linkeddata.es/dataset/lov> last accessed 28/03/24.

⁴¹ REgistry of REsearch data REpositories, <https://www.re3data.org> last accessed 28/03/24.

⁴² See <https://rdamsc.bath.ac.uk> last accessed 28/03/24.



7 Analysis

From the results presented in the previous Section, Table 8 summarises whether the metric tests used by each of the four tools considered could be used to test the FAIR4RS principles or whether additional work is required.

Table 8 – summary of whether the metrics used by each tool could be used to assess the corresponding FAIR4RS principle. The results indicate that the metrics used by the tool: Y - Yes; N - No, and P - Partially but more work is required to assess the FAIR4RS principle

FAIR4RS principle	howfairis	FAIR-Enough	FAIR-Checker	F-UJI
F1: Software is assigned a globally unique and persistent identifier.	Y	Y	Y	Y
F1.1: Components of the software representing levels of granularity are assigned distinct identifiers.	N	N	N	N
F1.2: Different versions of the software are assigned distinct identifiers.	Y	N	N	N
F2: Software is described with rich metadata.	P	Y	Y	P ⁴³
F3: Metadata clearly and explicitly include the identifier of the software they describe.	N	P	N	Y
F4: Metadata are FAIR, searchable and indexable.	N	P	N	P
A1: Software and its associated metadata are accessible by their identifier using a standardized communications protocol.	P	N	N	Y
A1.1: The protocol is open, free, and universally implementable.	Y	Y	Y	Y
A1.2: The protocol allows for an authentication and authorization	Y ⁴⁴	Y	Y	N

⁴³ It does not check for the FAIRness of the metadata or whether the metadata uses controlled vocabularies; we recognise that although FsF-I2-02M checks for “semantic resources are present in the metadata of an object”, this does not appear to be the intended target of Principle I2 (“Software includes qualified references to other objects.”), and should be instead tested as part of F2.

⁴⁴ Not tested directly but given the protocols that these repositories support authentication and authorisation and they provide mechanisms to allow for the credentials to be added it does provide an implicit test for A1.2.

procedure, where necessary.				
A2: Metadata are accessible, even when the software is no longer available.	P	Y ⁴⁵	N	Y
I1: Software reads, writes and exchanges data in a way that meets domain-relevant community standards.	N	N	N	N
I2: Software includes qualified references to other objects. ⁴⁶	N	P	P	P
R1: Software is described with a plurality of accurate and relevant attributes. ⁴⁷	P	N	N	P
R1.1: Software is given a clear and accessible license.	Y	Y	Y	Y
R1.2: Software is associated with detailed provenance.	N	N	Y	Y
R2: Software includes qualified references to other software.	P	N	N	N
R3: Software meets domain-relevant community standards.	Y ⁴⁸	N	P	Y

From this table, we can see that none of the tools examined can currently be applied directly to assess a repository's compliance with the full set of FAIR4RS principles. Because of the similarity between the FAIR principles and FAIR4RS principles, the assessment of the more generic principles can follow similar methods and there is not a great difference between these methods. However, Table 1 showed that some principles require domain-specific information to be properly assessed and require extensions to the corresponding metrics, their implementation as tests and the tooling that would run these. Tool developers cannot be expected to be domain experts, so the tooling must be extensible to support the inclusion of additional community-specified metrics and tests to perform checks by the FAIR4RS principles that require domain-specific input. An issue may arise where tests to assess the principles cannot be defined where there is no community consensus or agreed-upon standard. Where there is a lot of community fragmentation, this might lead to many tool flavours to support each sub-community.

⁴⁵ It looks for a persistence policy (persistencePolicy) in the metadata.

⁴⁶ In the FAIR4RS principles, I2 is derived from but slightly different from FAIR principle I3 - the number change may be confusing.

⁴⁷ Testing for the accuracy of the metadata is harder than testing for the existence of it.

⁴⁸ It meets the criteria of the OSSF badge and it also if it has been submitted to a relevant software distribution registry site through the presence of the relevant badge, e.g. CRAN, Pypi, npm, etc.

Tool development and sustainability is also bounded by funding, developer interest, the number of developers involved and other factors. At the time of writing, measuring activity by the number of commits, of the four tools examined two had not had a git commit within the last 18 months while the remaining two had had some activity within the previous six-month period as derived from Figure 3. Tool development continuity is an important factor to consider.

Another important factor is to consider how straightforward it might be to take an existing tool and use it as the basis to assess the FAIR4RS principles. In addition, an extensible framework is required which allows existing metric test implementations to accommodate new test implementations, and new or additional metrics and/or tests to be added to the existing framework and to be able to expose these through these same frameworks. Looking at how each of the tools examined might be extended:

- **howfairs** - although this is the only tool that at the time of the activity assessed the FAIRness of software, it does not do it based on the FAIR4RS principles. It is likely to require a lot of effort to accommodate the FAIR4RS framework.
- **FAIR-Checker** - It is not immediately obvious how one might extend the existing framework other than by going through the code. There is no explicit documentation as to how this process may be undertaken.
- **F-UJI** - documentation is provided as to how new metrics can be added and incorporated into the F-UJI framework⁴⁹.
- **FAIR-Enough** - additional Metrics Tests API URLs can be registered within an existing deployment by following the standard described by the FAIRMetrics Working group⁵⁰. The developers recommend using the fair-test Python library⁵¹. This provides a mechanism for adding or customising FAIR metrics tests. We therefore consider that FAIR-Enough offers the ability to add new metrics and tests to the existing tool.

⁴⁹ <https://github.com/pangaea-data-publisher/fuji>

⁵⁰ See <https://github.com/FAIRMetrics/Metrics> last accessed 28/05/24.

⁵¹ See <https://maastrichtu-ids.github.io/fair-test/> last accessed 28/05/24.

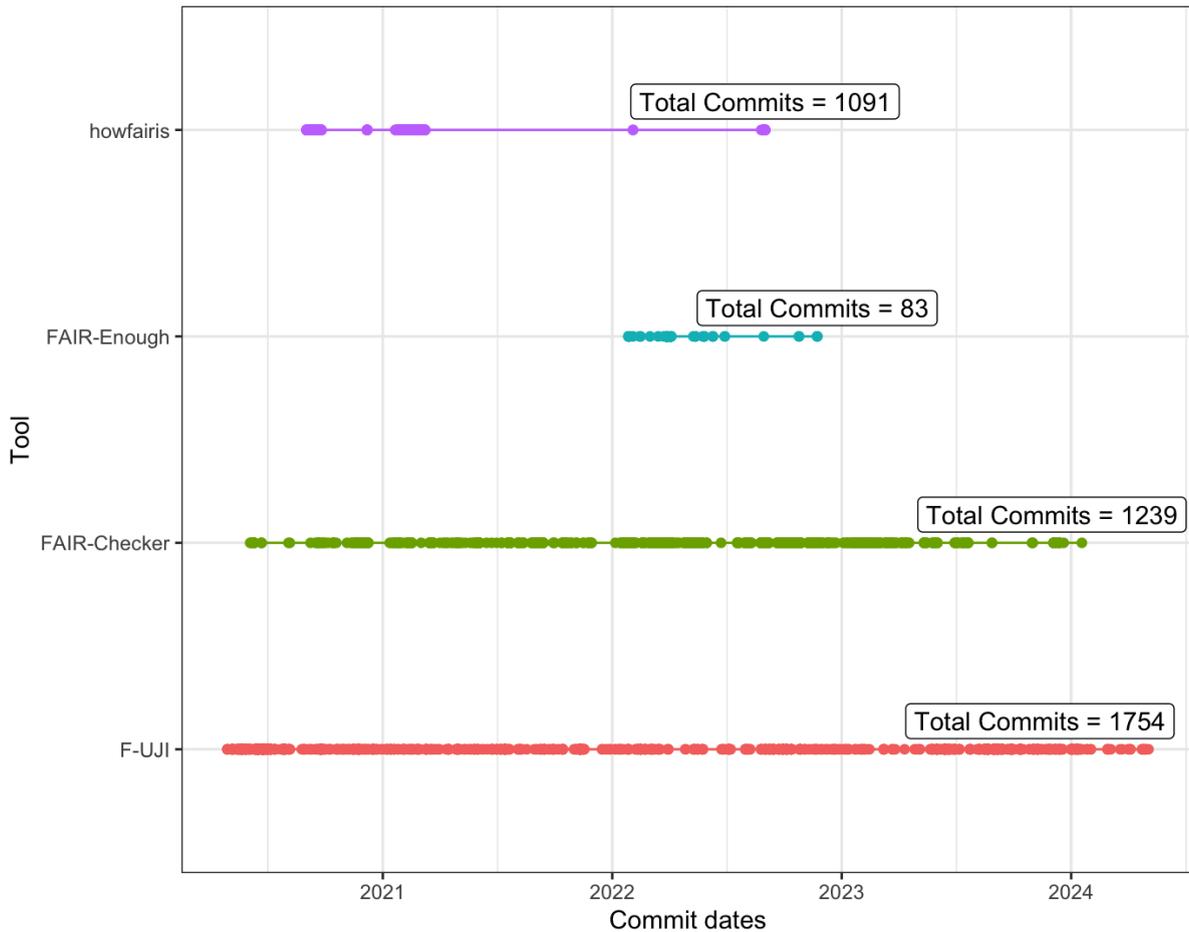


Figure 3: Level of activity for each tool measured by the number of git commits against time.

A set of FAIR4RS metrics tailored to a social sciences domain was proposed in Chue Hong *et al* (2023). Some of these metrics were then incorporated into F-UJI, see Moraw *et al* (2024). F-UJI was chosen for this purpose for two reasons: of all the tools investigated F-UJI was the tool that was being most actively developed; and the developers were also involved in the FAIR-IMPACT project so collaboration with the developers was straightforward.

The approaches taken by the various tools will also be a factor. An automated testing framework will, by necessity, need to make assumptions in order to simplify the testing: where will it find the expected metadata? Will all files be searched for metadata? Will only the root directory tree be searched, will the whole directory tree be scanned, or will search be limited to a codemeta.json file only? To make the process tractable, it is necessary for a tool to make simplifying assumptions which, in turn, will drive behaviour. The FAIR principles have no normative requirements but the desire to make autonomous assessment possible may normalise behaviour from users to improve their assessment results. It is also important for the tooling to be transparent about how it is making its assessment, for instance if a tool

does not recognise the addition of a metadata description file, e.g. `codemeta.json`, to a repository this could be frustrating for a user if they believe that these files are used as part of the assessment. Equally well, for an end user to be told what metrics or tests have failed is less useful than being told on how they can improve the FAIRness of their repositories. Finally, the tool developers need to ensure that their tools give consistent results for similar repositories. What would be useful for both tool developers and end users is for the FAIR community to provide exemplar repositories; tool developers could test their tooling against these, while end users could use them to see best practices applied and emulate them to become FAIRer citizens in publishing their data and/or research software.

8 Conclusions and next steps

This work examined some existing autonomous FAIR assessment tools and the metrics, tests, and assumptions they use for their suitability to perform autonomous assessments of software against the FAIR principles (as extended to software via the FAIR4RS principles). The FAIR data principles have a 5-year lead time over the FAIR4RS principles but the latter can build on the groundwork already done on the former. At the start of this work, there were no FAIR4RS metrics or autonomous assessment tools. Since then, a number of FAIR4RS metrics have been defined by the FAIR-IMPACT project, including versions applicable within a social science domain, and some of these metrics have been implemented within the F-UJI framework. It would be useful for tool developers and end-users for the FAIR4RS community to set up exemplar software repositories, so that tool developers could use them to test their FAIR4RS assessments and ensure consistency but more importantly for those wishing to adopt the FAIR4RS principles to emulate.



References

- Bahim, C., Casorrán-Amilburu, C., Dekkers, M., Herczog, E., Loozen, N., Repanas, K., Russell, K., & Stall, S. (2020). The FAIR data maturity model: An approach to harmonise FAIR assessments. *Data Science Journal*, 19. <https://doi.org/10.5334/dsj-2020-041>
- Chue Hong, N., Katz, Barker, Lamprecht, Martinez, Psomopoulos, Harrow, Castro, Gruenpeter, Martinez, Honeyman, Struck, Lee, Loewe, Werkhoven, van, Jones, Garijo, Plomp, Genova, ... Wg, R. F. (2022, May 24). *FAIR principles for research software (FAIR4RS principles)*. Zenodo. <https://zenodo.org/record/6623556#.YqCJTJNBwlw>
- Chue Hong, N., Breitmoser, E., Antonioletti, M., Davidson, J., Garijo, D., Gonzalez-Beltran, A., Gruenpeter, M., Huber, R., Jonquet, C., Priddy, M., Shepeherdson, J., Verburg, M., & Wood, C. (2023, October 27). *D5.2 - Metrics for automated FAIR software assessment in a disciplinary context*. Zenodo. <https://zenodo.org/doi/10.5281/zenodo.10047401>
- Devaraju, A., & Huber, R. (2021). An automated solution for measuring the progress toward FAIR research data. *Patterns*, 2(11), 100370. <https://doi.org/10.1016/j.patter.2021.100370>
- Devaraju, Huber, Mokrane, Herterich, Cepinskas, Vries, de, L'Hours, Davidson, & White, A. (2022, April 14). *FAIRsFAIR data object assessment metrics*. Zenodo. <https://zenodo.org/record/6461229>
- Emonet, V. & Dumontier, M. (n.d.) FAIR enough. <https://github.com/MaastrichtU-IDS/fair-enough> [accessed 2024-03-11]
- Gaignard, A., Rosnet, T., De Lamotte, F., Lefort, V., & Devignes, M.-D. (2023). FAIR-Checker: Supporting digital resource findability and reuse with Knowledge Graphs and Semantic Web standards. *Journal of Biomedical Semantics*, 14(1). <https://doi.org/10.1186/s13326-023-00289-5>
- Mons, B., Neylon, C., Velterop, J., Dumontier, M., da Silva Santos, L. O. B., & Wilkinson, M. D. (2017). Cloudy, increasingly FAIR; revisiting the FAIR Data guiding principles for the European Open Science Cloud. *Information Services & Use*, 37(1), 49–56. <https://doi.org/10.3233/isu-170824>



- Moraw, K., Antonioletti, M., Breitmoser, M., Chue Hong, N., Priddy, M. (2024). M5.6 - Practical tests for automated FAIR software assessment in a disciplinary context. <https://zenodo.org/doi/10.5281/zenodo.10890042>
- Plomp, E. (2020). Going digital: Persistent identifiers for research samples, resources and instruments. *Data Science Journal*, 19(1). <https://doi.org/10.5334/dsj-2020-046>
- Spaaks, J. H., Verhoeven, S., Tjong Kim Sang, E., Diblen, F., Martinez-Ortiz, C., Etuk, E., Kuzak, M., Werkhoven, B., Soares Siqueira, A., Saladi, S., & Holding, A. (2022). howfairis (0.14.2). Zenodo. <https://doi.org/10.5281/zenodo.7041464>
- Wilkinson, M. D., Dumontier, M., Aalbersberg, Ij. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., ... Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1). <https://doi.org/10.1038/sdata.2016.18>
- Wilkinson, M. D., Sansone, S.-A., Marjan, G., Nordling, J., Dennis, R., & Hecker, D. (2022, December 20). *FAIR assessment tools: Towards an “apples to apples” comparisons*. Zenodo. <https://zenodo.org/record/7463421>
- Wilkinson, M. D., Sansone, S.-A., Schultes, E., Doorn, P., Bonino da Silva Santos, L. O., & Dumontier, M. (2018). A design framework and exemplar metrics for FAIRness. *Scientific Data*, 5(1). <https://doi.org/10.1038/sdata.2018.118>



Appendix 1: howfairis tests

The following tests are used by howfairis to test each of the five Netherlands eScience Center and DANS recommendations.

1. Use a publicly accessible repository with version control

- Only checks repositories hosted at <https://github.com> or <https://gitlab.com>. It does not support self-hosted GitLab instances.

2. Include a License

- Checks for the availability of a LICENSE file at the project level page and parses the repository project page for a license reference.

3. Register your code in a community registry

- Check whether the code has been submitted to a standard software distribution site by looking for the inclusion of Shields.io badges in the README.md file. Specifically, it checks any one of the following:
 - The Astrophysics Source Code Library⁵² (ascl) badge.
 - The bintray badge - a service run by JFrog discontinued on 01/05/21 that used to facilitate the distribution of Java binaries⁵³.
 - The Conda badge - conda provides a package, dependency and environment management for several different computing languages and conda-forge⁵⁴ can be used to distribute software.
 - The CRAN badge - the Comprehensive R Archive Network (CRAN)⁵⁵ is the canonical way to distribute R packages.
 - The crates⁵⁶ badge - a Rust package registry.
 - The Maven⁵⁷ badge - Maven can manage a project's build, reporting and documentation from a central piece of information.
 - The npm⁵⁸ badge - JavaScript package management.
 - The pypi⁵⁹ badge - Python package index.
 - The rsd⁶⁰ badge - research software directory designed to show the impact research software has on research and society.

⁵² The Astrophysics Source Code Library <https://ascl.net> last accessed on 11/09/23.

⁵³ <https://jfrog.com/blog/into-the-sunset-bintray-jcenter-gocenter-and-chartcenter/>; accessed 25/01/24

⁵⁴ Conda-forge <https://conda-forge.org> last accessed on 19/09/23.

⁵⁵ CRAN <https://cran.r-project.org> last accessed 18/09/23.

⁵⁶ Crates <https://crates.io> accessed 21/09/23.

⁵⁷ Maven <https://maven.apache.org> last accessed 21/09/23.

⁵⁸ Npm <https://www.npmjs.com> last accessed 21/09/23.

⁵⁹ Pypi <https://pypi.org> last accessed 21/09/23.

⁶⁰ Research software directory <https://research-software-directory.org> last accessed 21/09/23.

- The GitHub marketplace⁶¹ badge - demonstrates that your code is available through the GitHub marketplace. The marketplace allows developers to improve and extend GitHub workflows
4. **Enable citation of the software**, by checking for any one of the following:
- Check for a CITATION file in the repository.
 - Check for a CITATION.cff file in the repository.
 - Look for a codemeta.json file produced by the CodeMeta generator⁶² in the repository.
 - Look for the presence of a Zenodo badge.
 - Look for a “.zenodo.json” file which allows developers to override the default metadata obtained using the GitHub API⁶³.
5. **Use a software checklist**
- Checks for the core infrastructure best practice badge⁶⁴.

Compliance with recommendations 3 and 5 are obtained through the parsing of a README in markdown or restructure text format (these are case sensitive, and the file must be stored in a standard location, as defined by the tool; e.g. the top-level directory of the repository); the tool looks for any one of the Shields.io badges⁶⁵ in the contents of the README files. At the time of writing, howfairis only performs its tests against code hosted on either <https://github.com> or <https://gitlab.com>; no other software repository deployments, including local deployments of the GitLab infrastructure, are supported.

⁶¹ GitHub market place <https://github.com/marketplace> last accessed 21/09/23.

⁶² CodeMeta generator <https://codemeta.github.io/codemeta-generator> last accessed on 21/09/23.

⁶³ See <https://developers.zenodo.org/#other-questions-on-harvesting> last accessed on 21/09/23.

⁶⁴ Core infrastructure badge <https://www.bestpractices.dev> last accessed on 21/09/23.

⁶⁵ Badges <https://shields.io> last accessed 07/09/23.