FAIRSECO: An Extensible Framework for Impact Measurement of Research Software

Deekshitha*†‡, Siamak Farshidi*, Jason Maassen†, Rena Bakhshi†, Rob van Nieuwpoort†‡, Slinger Jansen*

*Department of Information and Computing Sciences, Utrecht University, Utrecht, the Netherlands
{d.deekshitha,s.farshidi,slinger.jansen}@uu.nl
†Netherlands eScience Center, Amsterdam, the Netherlands
{j.maassen,r.bakhshi,r.vannieuwpoort}@esciencecenter.nl
‡University of Amsterdam, Amsterdam, The Netherlands

Abstract—The growing usage of research software in the research community has highlighted the need to recognize and acknowledge the contributions made not only by researchers but also by Research Software Engineers. However, the existing methods for crediting research software and Research Software Engineers have proven to be insufficient. In response, we have developed FAIRSECO, an extensible open source framework with the objective of assessing the impact of research software in research through the evaluation of various factors. The FAIRSECO framework addresses two critical information needs: firstly, it provides potential users of research software with metrics related to software quality and FAIRness. Secondly, the framework provides information for those who wish to measure the success of a project by offering impact data. By exploring the quality and impact of research software, our aim is to ensure that Research Software Engineers receive the recognition they deserve for their valuable contributions.

Index Terms—FAIR, research software engineering, software impact measurement, software citations

I. INTRODUCTION

The FAIR4RS Working Group [1] defines research software (RS) as source code files, algorithms, scripts, computational workflows and executables created during the research process or for a research purpose. In contrast, other types of software, such as operating systems, libraries, dependencies, and packages are not explicitly created for research but are employed in research activities.

Recent studies [2], [3] have shown that 33% of international research produces new code and 90-95% of UK and US researchers use and acknowledge RS as important for their research. In many research projects, Research Software Engineers (RSEs) closely collaborate with researchers to understand their challenges and to develop RS that helps to provide the answers to their research questions [4]. Reusing (parts of) RS is beneficial to the research community. It saves time and effort and encourages collaboration and good coding practices. By encouraging RS reuse, the developers increase their work's impact and foster a collaborative environment that drives scientific progress [5].

Two factors contribute to the success of RS reuse. First, like research data, the RS should be made FAIR [6], [7]. Following the FAIR principles increases the potential for the RS to be found and reused by other researchers, as well as encouraging good coding practices and enabling software citation.

Second, when reusing RS, proper credit should be given to the developers. Similar to scholarly papers and data sets, RS should get recognition through citations. The current lack of consistent software citation makes it difficult to measure the reuse and impact of RS and creates challenges in giving credit to RSEs for their contributions [8]. Properly citing RS will assist the authors in obtaining financial support for further software development, thereby improving the RSEs career paths.

Many guidelines and tools exist that help developers improve the FAIRness, quality, and citeability of their code [9], [10], [34], [35], [36]. Currently, however, there is no single framework RSEs can use to get an overview of the FAIRness, quality and impact of their RS. Moreover, Gomez-Diaz and Recio [9] and Istrate et al. [11] argue that there is currently no sufficient method to evaluate the quality and impact of RS. To address this issue, we have created the FAIR Research Software Ecosystem (FAIRSECO) framework [37]. The FAIRSECO framework is designed to combine different RS metrics under one extensible framework. It combines data from many existing tools that provide information on license conflicts, dependencies, method-level code-reuse, bibliometric analysis, citation metadata, FAIRness aspects, code indexing, and Software Bill of Materials (SBOM) generation to generate a concise overview of the quality, FAIRness, and impact of RS. FAIRSECO then combines the output of these existing tools into two scores, one for quality and one for impact. This enables RSEs to quickly gain insight into the recognition their software already receives and provide suggestions on how they can improve the quality and potential impact of their software.

The idea of the FAIRSECO framework has been introduced in a conceptual short paper [12]. However, that short paper focuses only on impact measurement through method reuse. While method reuse is important, we have evolved our FAIR software ideas and significantly broadened our scope.

The contribution of current work is twofold:

- **Integration of quality, FAIRness and impact measurement tools into a single extensible framework**: The FAIRSECO framework consolidates a comprehensive

collection of tools designed to measure various aspects of RS quality, FAIRness, and impact into a single, unified framework. This provides researchers with a convenient and efficient solution for measuring and assessing these aspects of their RS.

- **Quantifying the quality and impact of RS based on key features**: We provide a novel Quality Score ($S_{quality}$) for assessing the quality of RS based on key features, incorporating FAIRness, license violations, maintainability, and documentation as factors in the equation. By considering these aspects together, we provide a robust and objective method for systematically measuring and quantifying RS quality. Similarly, an Impact Score (S_{impact}) is introduced based on three factors: the number of citations, the number of reused methods, and the Quality Score.

We organize this paper as follows. First, we show what other tools are available and how they compare to the FAIRSECO framework in the related work Section II. This includes a brief description of each tool used by the FAIRSECO framework. Next, a full description is provided of the FAIRSECO framework in Section III. The FAIRSECO framework is then demonstrated in action in Section IV by its application to an existing RS project. The evaluation of five RS projects using FAIRSECO is presented in Section V. Finally, the paper concludes with a summary and discussion of the FAIRSECO framework infrastructure and a set of steps to take in the near future in Section VI.

II. RELATED WORK

We observed a shortage of tools and frameworks RSEs could use to measure their impact on the RS ecosystem. Therefore, we aimed to build an accurate framework for evaluating software quality and measuring the effect of RS. To this end, we reviewed FAIRSECO framework and existing tools focusing on identification of license violation (license checking), examining the dependencies of a software project for outdated components or vulnerabilities (dependency checking), method level checking of code reuse (method-level checking), analysis of citation, impact related features (bibliometric analysis), checking presence of citation (citation file checking), checking compliance to the FAIR principles (FAIRness aspects), building the source code index (code indexing), and building a list of all the components and dependencies used in a software project (SBOM generation).

A. Tools of FAIRSECO

We integrated several tools and external data sources into the FAIRSECO framework, namely, the tools HOWFAIRIS, TORTELLINI, and SEARCHSECO.

HOWFAIRIS is a Python package to analyze a GitHub or GitLab repository's compliance with the recommendations given on fair-software.eu [34]. The HOWFAIRIS generates a FAIRness report with the help of FAIRTALLY [38]. Based on this report, users identify areas where improvements are needed to make their repository more compliant.

The Netherlands eScience Center has developed a tool called TORTELLINI [36], a GitHub action that checks for any licensing issues in a given software, such as incompatible licenses in the software's dependencies.

SEARCHSECO [13] is a hash-based index for code fragments that enables searching for source code at the method level in the global software ecosystem [14]. SEARCHSECO supports a number of languages and deals with multi-language projects.

Table I summarizes the functionality of each of the tools described above. HOWFAIRIS focuses on citation file checking and FAIRness aspect but lacks support for dependency checking and other tasks. Similarly, SEARCHSECO only performs method-level checking and code indexing, and TORTELLINI only supports license checking. Finally, FAIRSECO framework performs all listed tasks, except dependency checking. In particular, FARSECO performs license checking, method-level checking, citation file checking, FAIRness aspect checking, code indexing with the help of HOWFAIRIS, TORTELLINI, and SEARCHSECO. However, FAIRSECO is limited to the Github repositories.

B. Related platforms

GitHub, the largest open-source repository, uses metrics such as stars, watchers, and forks (as a part of its Bibliometric Analysis) to measure the impact of software. The star system in GitHub functions similarly to the 'like' button in social media platforms, indicating user interest or support for a project or being used as a bookmark. However, there is a lack of comprehensive and well-founded empirical research to determine the exact meaning and practical implications of 'starring a project' in GitHub [15]. The credibility of the star metric on GitHub is questionable, as it is possible to create counterfeit or bot accounts that are used to give an excessive number of stars to a project. However, it performs dependency checking through dependency bots, and if citation file is present, it provides options for citing repository.

LIBRARIES.IO [39] is a platform designed to assist RSEs in discovering new open-source libraries, modules, and frameworks and keep track of the ones they rely on. Its primary objective is to enhance software quality by addressing three critical issues: discovery, maintainability, and sustainability. LIBRARIES.IO performs dependency and license checking but does not support the other tasks.

Depsy [16] measures impact of RS through software citations such as imports by other software and through references to software from papers. It is, however, limited to the RS available via the Python packages repository PyPI and the R packages repository CRAN.

The tooling and services known as Software Heritage Graph (SWH-GRAPH) [17] offer quick access to the graph representation of the Software Heritage Archive, an archive that attempts to collect as much software as possible. Currently, it can claim to have the largest collection of source code. These services are collectively called SWH-GRAPH and operate based on an in-memory, compressed representation of the Software Heritage

TABLE I

COMPARISON TABLE: THIS TABLE HIGHLIGHTS A COMPARATIVE ANALYSIS BETWEEN FAIRSECO AND OTHER TOOLS, EMPHASIZING SPECIFIC IMPACT METRICS.

Tools	Feature							
	License checking	Dependency checking	Method level checking	Bibliometric Analysis	Citation file checking	FAIRness Aspect	Code indexing	SBOM Generation
HOWFAIRIS	✗	✗	✗	✗	✔	✔	✗	✗
SEARCHSECO	✗	✗	✔	✗	✗	✗	✔	✗
TORTELLINI	✔	✗	✗	✗	✗	✗	✗	✗
LIBRARIES.IO	✔	✔	✗	✗	✗	✗	✗	✗
SWG-GRAPH	✗	✗	✗	✗	✗	✗	✔	✗
RSD	✗	✗	✗	✔	✔	✔	✗	✗
DEPSY	✗	✗	✗	✔	✗	✗	✗	✗
GITHUB	✗	✔	✗	✔	✔	✗	✔	✗
GRIMOIRELAB	✗	✗	✗	✔	✗	✗	✔	✗
SIRGRID	✗	✔	✗	✗	✗	✗	✗	✗
SQAAAS	✔	✗	✗	✔	✔	✔	✗	✗
FAIRSECO	✔	✗	✔	✔	✔	✔	✔	✔

Merkle DAG. SWH-GRAPH is limited to code indexing, as it does not perform any of the other listed tasks.

The RESEARCH SOFTWARE DIRECTORY (RSD) [40] is designed to show the impact of RS on scientific community. Its primary objective is to promote RS reuse and encourage proper RS citation to ensure researchers and RSEs get credit for their work. The RSD, by default, is configured to gather RS data from various platforms such as GitHub, Gitlab, OR-CID, Research Organization Registry, Zenodo, DataCite, and Crossref. The RSD performs bibliometric analysis, citation file checking, and FAIRness aspect checking.

Software development analytics platforms and toolsets such as GrimoireLab [18], SIGRID and SQAAAS [19] take a modular approach similar to FAIRSECO employing several tools that collect metadata to compile a quality report of software for their developers. However, none of these tools explicitly consider impact of RS.

SQAAAS focuses on improving RS quality by offering RSEs ready-to-use continuous integration and continuous development pipelines that align with the principles of Open Science [41]. It performs license checking, citation file checking, reports on quality of RS. It is integrated with FAIR data assessment tools which follows the FAIRsFAIR route [20].

SIGRID performs only dependency checking, and the rest of the tasks such as FAIRness aspect, citation file checking, biblometric analysis, license checking are not applicable because the scope of SIGRID is on industrial software and its compliance to the industrial standards.

GRIMOIRELAB is an open-source toolkit focusing on data collection, indexing and storage in the GRIMOIRELAB database for potential analysis of software development. It provides very powerful and flexible basis for analysis, but requires an additional efforts and expertise to extract and display relevant information.

Thus, FAIRSECO offers a unique combination in terms of the considered features for assessing impact and quality of RS.

III. THE FAIRSECO FRAMEWORK

A. The FAIRSECO Extensible Software Architecture

Figure 1 depicts the architecture of the FAIRSECO framework and its components. The framework consists of five primary components: a Code analyzer, Data collection unit, Impact calculation module, Report, and Artifacts module. All these components are interconnected with a FAIR artifact generator. The FAIR artifact generator will receive input from the Code analyzer, Data collection unit, and Impact calculation module and generates Reports and Artifacts as output.

The FAIRSECO integrates different sources of information and tools to support RSEs in *becoming more FAIR*, *improving and measuring RS quality*, and *measuring the impact of their RS*. The framework uses the RS's repository such as GitHub as input for checking FAIRness and quality and uses other data sources for measuring impact and quality.

B. Collecting data

The data collection unit consists of a crawler and an API call component to collect citation data and code reuse details from various sources such as Semantic scholar [42], OpenAlex [21], and SEARCHSECO.

Besides generating a report, the FAIRSECO framework also creates artifacts, such as:

- **SBOM** SBOM is a structured list of third-party software components and libraries included directly or indirectly in the code [43]. It contains information about open-source licenses and version numbers. FAIRSECO will produce an SBOM for the RS using the tool named SBOM.
- **TORTELLINI Results** A detailed report is generated specifically for identifying license violations within the RS. This report provides comprehensive information about the licenses of packages used in the RS, including the primary packages and their dependencies. It includes details such as each package's specific license types and version information. By examining this report, RSEs identify any instances where the RS may violate license agreements, ensuring compliance with licensing
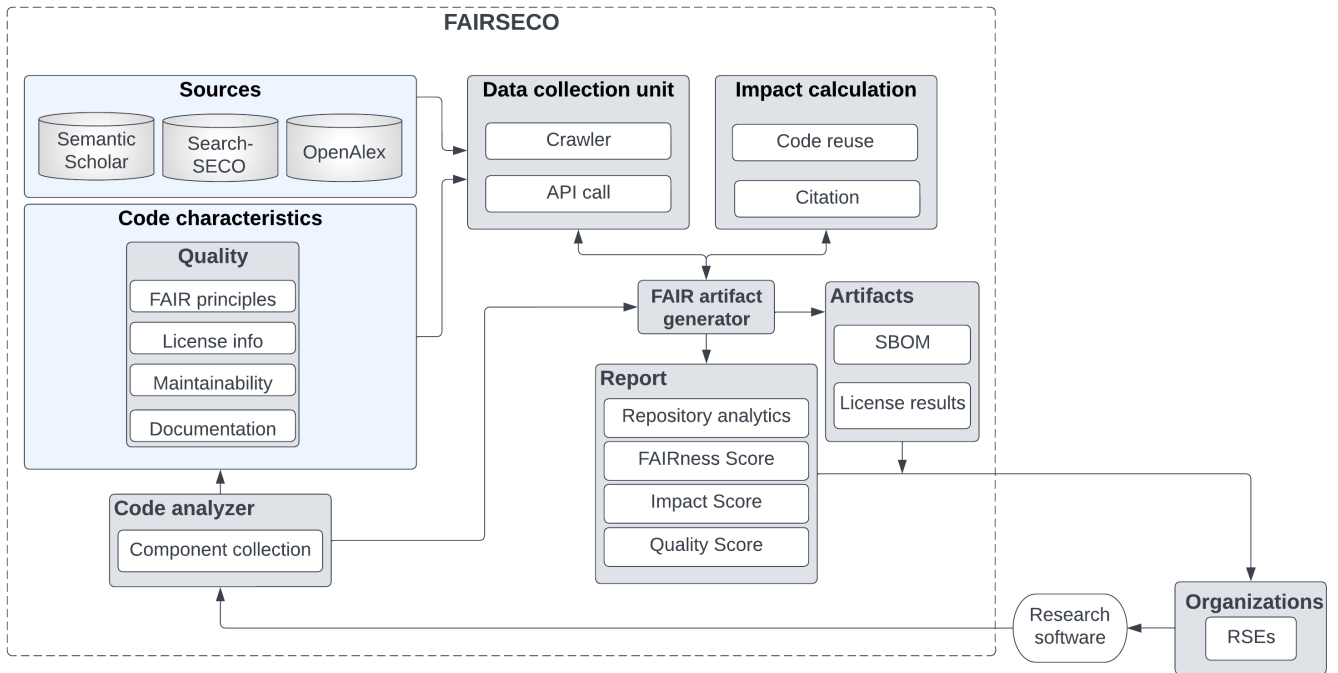
Fig. 1. FAIRSECO architecture consists of five components: 1) Data collection unit gathers data from different data sources, 2) Impact calculation performs an impact calculation, 3) Code analyzer analyzes the code of the RSE project, 4) Report generates final report related to Quality and Impact, and 5) Artifacts generates two artifacts, such as SBOM, and License results for the RS.

requirements and addressing any potential legal or ethical concerns related to the software's usage and distribution.

C. Becoming more FAIR

The FAIRSECO framework assesses the FAIRness score (S_{fair}) of RS based on five recommendations for FAIR software [22], and it is calculated using HOWFAIRIS tool. Explanations for these recommendations are given below:

R1) **Use a publicly accessible repository with version control**: The initial recommendation focuses on verifying that the RS utilizes a publicly accessible repository with version control. This ensures widespread accessibility and encourages transparency. By employing a version control system, RSEs can easily monitor and manage modifications made to the RS, facilitating effective collaboration and traceability.

R2) **Add a license** It is important to emphasize that an appropriate license must accompany the RS. Without a license, even if the RS is publicly available on platforms such as GitHub, the absence of legal permissions makes it impossible for anyone to use it.

R3) **Register your code in a community registry** This recommendation suggests registering RS in a community registry such as Zenodo, CodeOcean, FigShare, Software Heritage Archive, etc. Registering RS makes it easier for others to find it, particularly through the use of search engines such as Google.

R4) **Enable citation of the software** Citation is an integral part of scientific accountability and reproducibility. Cita-

tion helps RSEs be recognized for their work. Citation File Format (CFF) [35] is specifically designed to enable the citation of software.

R5) **Use a software quality checklist** This recommendation proposes examining quality checklists that have been included on fair-software.eu[1]

The S_{fair} is calculated by combining these recommendations, the RS gets a maximum score of 5, and each recommendation contributes a maximum score of 1 (cf. [34], [38]). We will provide a more detailed explanation in the below subsections on how these scores are calculated.

D. Measuring RS Quality

The code analyzer in the FAIRSECO framework architecture evaluates the quality of RS. It employs various techniques to analyze the RS, encompassing the evaluation of FAIRness [24], identification of license violations, examination of software maintainability, and checking the presence of documentation. Using $S_{quality}$, RSEs evaluate their project's development and identify improvement areas. The following sections describe the main features contributing to $S_{quality}$.

1) License violation or information: License violations have significant consequences for violators and open-source communities [25]. These violations damage the reputation

---

[1]fair-software.eu is a website that promotes FAIR practices in RS development by offering five key recommendations for FAIR software [23]. It aims at raising awareness about the importance of making RS FAIR. It provides guidelines and resources for researchers and RSEs to adopt FAIR software principles and best practices.

of a project and discourage further contributions from the community, which have negative long-term effects [26].

We argue that researchers and RSEs should carefully review the license associated with any open-source software before using it and ensure that they agree to the terms and conditions of that license. This practice can foster the perpetual expansion and prosperity of the open-source ecosystem.

We utilized TORTELLINI for predicting license violations; the FAIRSECO framework helps users become more aware of potential license violations and the potential problems that arise.

2) Maintainability: Software maintainability refers to the ease with which the operations, such as the addition of new features, obsolete code deletion, and error correction, are carried out [27]. It is an aspect of software development that consumes a significant portion of the overall project cost. However, measuring maintainability during the early stages of software development aids in better planning and optimal resource utilization [28].

The FAIRSECO framework analyzes open and closed issues on GitHub to assess maintainability. It calculates maintainability by calculating the percentage of issues that have been closed since the date on which the software was added to GitHub. RSEs utilize the maintainability score as a guideline to improve their development process and address current open issues more efficiently.

3) Documentation: To be usable, open-source software needs to be sufficiently documented. This helps users understand the functionality and usability of the software. GitHub and Zenodo are platforms that typically include a README file and documentation for each software, giving users information about the software.

The FAIRSECO framework checks whether the RS is adequately documented to ensure users access the necessary documentation. Therefore, FAIRSECO assigns a maximum score of 100 if the RS includes both README and documentation.

4) Reporting on Quality: The code analyzer generates reports that summarize the analysis results. These reports provide RSEs with actionable feedback on the quality of their code. The reports highlight areas that require improvemnt, encompassing aspects such as FAIRness, license information or violations, maintainability, and documentation. By reviewing these reports, RSEs make informed decisions to enhance the overall quality of their RS.

Finally, we have formulated an equation to determine the $S_{quality}$, incorporating factors such as FAIRness, license violations, maintainability, and documentation:

$$S_{quality} = \frac{S_{fair} \cdot W_f + S_l \cdot W_l + S_m \cdot W_m + S_d \cdot W_d}{W_{total}} \quad (1)$$

The all weights, namely, the FAIRness weight W_f, license weight W_l, maintainability weight W_m, and documentation weight W_d are four distinct constant assigned to each factor. The total weight W_{total} is calculated by summing up these weights $W_{total} = W_f + W_l + W_m + W_d$. Currently, $S_{quality}$ is

determined using weighted criteria; however, future work will investigate the justification behind these weights. The scores used in (1) are calculated as follows:

- The S_{fair} is calculated by multiplying the HOWFAIRIS tool's output (0-5) by 20.
- The license score S_l is calculated using the license information provided by the TORTELLINI tool as

$$S_l = F_l^{\log_2(1+N_l)} \cdot 100 \quad (2)$$

where F_l is the fraction of licenses that are not violated, and N_l is the total number of licenses.

- The maintainability score S_m is calculated as follows:

$$S_m = \frac{100 \cdot C_i}{N_i} \quad (3)$$

where C_i is the number of closed issues, and N_i is the total number of issues.

- The documentation score S_d is determined based on the presence of README, and documentation files. The score is assigned a value of 50 if there is a documentation directory and another 50 if there is a README file. These scores are then combined to obtain the overall documentation score.

Thus, a score between 0 and 100 will be assigned to the RS based on Equation (1). The calculations described above represent our current FAIRSECO approach, considered a starting point. However, we plan to enhance the formulas by incorporating more detailed and higher-quality metrics. The framework is designed to be sufficiently generic to accommodate these expansions.

E. Measuring RS Impact

One of the largest challenges in providing insight into RS impact is where and how to report this data. There are indicators such as GitHub stars, inclusions of a package in dependency trees, citations to articles, mentions in news items, the number of issues posted by non-community members, etc. Gathering and interpreting these data is challenging. For FAIRSECO, currently, we have implemented two impact measures: citations and code reuse. Furthermore, researchers would likely hesitate to use poor-quality RS that introduces bugs, performance and maintenance issues, etc., because of potentially erroneous results produced by the RS and higher costs for fixing its technical debt [29]. Thus, we consider the quality of RS as an additional factor that influences S_{impact}, along with citations and code reuse. This intuition is in line with recent efforts of policymakers regulating long-term plans for maintaining RS [30].

1) Citation: In academic writing, the practice of citing relevant literature to establish connections with other works in the field is considered essential and common practice [31]. Recognizing the significance of this practice, FAIRSECO offers valuable assistance to its users by facilitating the rapid and efficient retrieval of citation data for their RS. This is accomplished by gathering information from APIs of the open catalogues of scientific publications (in particular, OpenAlex

and Semantic Scholar), enabling users to conveniently obtain the necessary citations for their RS. Moreover, it collects information about the scientific fields in which the RS is cited. It generates a radar citation graph [32] illustrating how the RS is connected across different scientific fields. A screenshot is provided in Figure 3.

2) Code Reuse: For this, the FAIRSECO framework utilizes SEARCHSECO: it feeds a repository link to SEARCHSECO, which then parses all the source code from the RS project and calculates a hash for an abstract version of each method in the work. Subsequently, SEARCHSECO looks in its database for occurrences of the method in the worldwide software ecosystem. If found, there are two possible outcomes:

1) (A fraction of) the RS is being reused by other parties, showing its impact. No further action is needed if the other software uses the RS's intellectual property correctly.
2) The RS project is reusing other software. While this does not lead to impact, it does provide an opportunity to ensure that the code in the RS project is used correctly with correct references and that the reused methods do not contain any vulnerabilities, which is a feature of SEARCHSECO.

3) Reporting on Impact: In addition to citations and code reuse, we include $S_{quality}$ as a factor influencing the Impact score. Similarly to the $S_{quality}$, the S_{impact} is determined by applying constant weights to factors such as the number of citations N_c, code reuse in other projects (N_r) as well the $S_{quality}$ obtained from Equation (1):

$$S_{impact} = \frac{N_c \cdot W_c + N_r \cdot W_r + S_{quality} \cdot W_q}{W'_{total}} \qquad (4)$$

where citation weight W_c, reused weight W_r, quality weight W_q are constants. W'_{total} is calculated by summing up these weights: $W'_{total} = W_c + W_r + W_q$.

Note that these constant weights can be changed after an expert evaluation and the impact measurement metric can be extended with more elaborate metric.

IV. THE FAIRSECO FRAMEWORK IN ACTION

This section overviews how RSEs utilize FAIRSECO to generate reports on FAIRness, Quality, and Impact. Generally, RSEs will want to use the FAIRSECO framework every couple of months or when they wish to provide a report on the impact of their repository. The history feature of the FAIRSECO infrastructure supports this, as the RSEs show the improvements they made (or did not make) regarding the impact and citation of their software. For a complete manual to operate the FAIRSECO GitHub Action, please see the documentation at README.

RSEs face challenges in monitoring the usage of their software, tracking academic papers that reference it, and identifying software built upon it. With the FAIRSECO framework, RSEs easily access information on license violations, impact, citation details, and the $S_{quality}$ of their RS. This eliminates the need to identify such details, thereby saving time and effort

manually. The report generated by the FAIRSECO framework increases the chances of other researchers citing the work of RSEs.

To illustrate the functionality of the FAIRSECO framework in a RS context, we have chosen the MCFLY tool [44] from GitHub. The following are the pages or tabs within the FAIRSECO framework, each with its respective explanation:
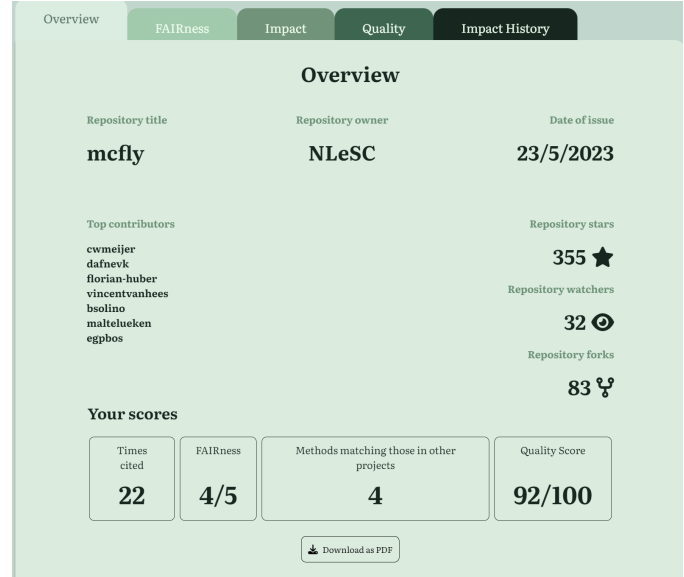

Fig. 2. Overview tab: This screenshot displays a summary of the FAIRSECO report, including the number of citations, S_{fair}, matching methods, $S_{quality}$, and several metrics from GitHub. For a more detailed understanding of the $S_{quality}$, please refer to Section IV

1) Overview tab: In Figure 2, we provided a screenshot of the "Overview" tab of the framework, which displays various details about the evaluated tool. These details include the repository title, owner, issue date, number of stars, watchers, and forks. Detailed explanations of each of these terms are given below:

1) **Repository title**: Repository title refers to the repository's name, which is evaluated using the FAIRSECO framework.
2) **Repository owner**: The repository owner is the name of the individual or organization who owns the repository on GitHub. In case the repository has been forked, it shows the user's name who forked the primary repository.
3) **Date of issue**: The issue date refers to when the repository was assessed using the FAIRSECO framework.
4) **Repository stars**: The Repository stars refer to the number of stars the repository receives on GitHub.
5) **Repository watchers**: The number of users who have added the repository to their watchlist on GitHub is added to the Repository watchers list.
6) **Repository forks**: The Repository forks refers to the number of times the repository has been forked on GitHub.

Moreover, the FAIRSECO framework gives further information, including citation counts, FAIRness value, method

matching with other projects, and $S_{quality}$ for the evaluated repository. Detailed explanations of these terms are provided in the FAIRSECO framework's next tab. Additionally, there is an option to download this report as a PDF.

2) Citation tab: An overview of the research papers that have cited RS is presented in the citation tab. The list of papers includes their title, author's names, description indicating the source from which the paper is collected, publication date, DOI, and a link to the paper.

3) FAIRness: The report on FAIRness is generated by analyzing the results of the HOWFAIRIS tool, and it is presented in the FAIRness tab. For this example tool, it has received a S_{fair} of 4 out of 5, because it did not meet all the quality criteria outlined in the 5 recommendations for FAIR software.

4) License violation: The FAIRSECO employs the TORTELLINI tool to provide a comprehensive account of license violations, informing the original author of the repository or the user about any infractions. The license violations tab displays the information regarding license violations.

5) Impact: As shown in Figure 3, the Impact section of FAIR-SECO displays various statistics about using the software in scientific research. It includes information about the frequency of citations the software has received across different fields, visually represented in a radar chart showcasing these fields' distribution. Additionally, it provides information on the most significant scientific paper that cited the software based on the number of citations and the journal's reputation.

Fig. 3. Impact tab: The Impact tab screenshot presents an overview of the RS MCFLY impact. It has been cited a total of 4 times, with two of those citations coming from different fields. Additionally, the screenshot includes a radar graph that illustrates the software's citation frequency across various fields. For a more detailed understanding of this tab, please refer to Section IV.

6) Quality Score ($S_{quality}$): The $S_{quality}$ tab is illustrated in Figure 4. The FAIRSECO assigns a $S_{quality}$ to each project and explains how this score was calculated. Four key metrics are considered, including the percentage of adherence to FAIRness principles, the extent to which the software complies with the license requirements, the percentage of closed issues on GitHub, and the presence of README and documentation

files. This score doesn't reflect the actual quality of a project's code but rather indicates how well the project follows the best practices of software engineering. The example repository received a $S_{quality}$ of 92% out of 100 because it did not pass the quality checklist. Thus, it receives a FAIRness value of 4 out of 5.
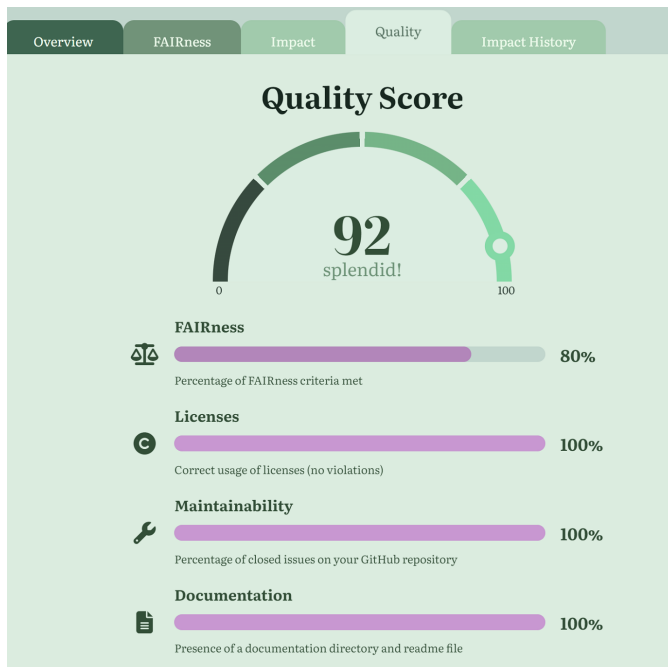

Fig. 4. Quality Score tab: This screenshot shows an overview of the generated FAIRSECO $S_{quality}$. For a complete explanation of the $S_{quality}$, see Section IV.

7) Impact History: The impact history tab provides a historical overview of the impact of RS by displaying information from previous runs of the FAIRSECO, if available. This feature allows RSEs to track how the impact of their software has evolved. The overview section shows the changes in $S_{quality}$, number of citations, reused methods, and S_{fair} since the last run. Ideally, RSEs will observe an increase in these metrics over time. The graph below visually represents this data, including runs before the most recent one.

V. EVALUATION

Table I compares five tools based on their performance concerning features such as Recommendation 1 (R1), Recommendation 2 (R2), Recommendation 3 (R3), Recommendation 4 (R4), Recommendation 5 (R5), license violation, maintainability, and documentation. The FAIRSECO framework calculates a $S_{quality}$ for each evaluated tool based on these features, also provided in the table. Each of these tools such as KERNEL TUNER [45], SPEC2VEC [46], ESMVALTOOL [47], GPT INDEX [48], and CFF-CONVERTER-PYTHON [49] were evaluated using the FAIRSECO tool. Among the five tools compared, KERNEL TUNER and SPEC2VEC met the FAIRness criteria and had a higher $S_{quality}$ than the other tools. KERNEL TUNER stands out in its FAIRness as it is registered in

TABLE II

Evaluation table: This table compares selected tools using the FAIRSECO framework. See Section III-C for the definition of R1-R5.

Tools	Feature									
	R1	R2	R3	R4	R5	FAIRness	Licenses	Maintainability	Documentation	Quality Score
Kernel Tuner	✔	✔	✔	✔	✔	100%	100%	100%	50%	94%
Spec2Vec	✔	✔	✔	✔	✔	100%	100%	100%	50%	94%
ESMValTool	✔	✔	✔	✔	✗	80%	100%	100%	50%	86%
GPT index	✗	✔	✔	✔	✗	60%	100%	100%	100%	85%
CFF-Converter-Python	✔	✔	✔	✔	✔	100%	19%	100%	100%	78%

TABLE III

Impact Score Table: Impact Score table shows the influences of various factors on the overall Impact Score

Tools	Feature			
	# citations	Reused in other Projects	Quality Score	Impact Score
Kernel Tuner	17	5	94%	60%
Spec2Vec	4	5	94%	56%
ESMValTool	40	4	86%	61%
GPT index	0	3	85%	50%
CFF-Converter-Python	0	5	78%	46%

Zenodo with a DOI, making it easily findable. Additionally, it is publicly accessible through GitHub and downloaded using standard protocols such as HTTPS. Moreover, Kernel Tuner has citation and license files, further enhancing its FAIRness. Therefore, Kernel Tuner scores a perfect 5 out of 5 regarding FAIRness criteria. According to our analysis, it does not violate any licenses. Kernel Tuner demonstrates effective maintenance by actively closing issues on GitHub, which is properly documented through documentation files and a README file. The ESMValTool does not meet the quality checklist mentioned in 5 recommendations for FAIR software and receives a lower $S_{quality}$ than Kernel Tuner and Spec2Vec. GPT index scored lower in the R1 and R5 categories but performed well in R3, R4 and R5. CFF-Converter-Python scored relatively low in the $S_{quality}$. Still, it performed well in all five features, scoring less due to issues with the version of the PyPI package, resulting in a license violation.

In summary, Kernel Tuner and Spec2Vec stand out as highly reliable and FAIR tools, since while ESMValTool, GPT index, and CFF-Converter-Python show strengths in specific areas but may require some attention to enhance their overall performance.

Table III gives the S_{impact} for selected tools. As per Equation (4), S_{impact} is calculated based on three factors: the number of citations, the number of methods reused, and the $S_{quality}$. For example, the tool Kernel Tuner has 17 citations, has been reused in 5 other projects, and has a $S_{quality}$ of 94%. Using Equation (4), the S_{impact} for Kernel Tuner is 60%.

The FAIRSECO framework provides a comprehensive report on the quality and impact of RS on research. RSEs utilize this report to identify areas where improvements are needed and make necessary changes to their software.

VI. Conclusion and Future Work

RS is becoming increasingly valued in the research ecosystem, leading to the evolution of international and national policy practices to reflect its importance. The FAIRSECO framework, with its collection of tools, is designed to report on the quality and impact of RS. FAIRSECO framework automatically measures the FAIRness of the RS using five recommendations for FAIR software. This is achieved by conducting several checks to ensure that the RS is registered in a community registry, making it available in a public repository, using a quality checklist, encouraging the use of the CFF to facilitate appropriate citation, and including a license file to enable reuse by other researchers.

To assess the quality of the software, the FAIRSECO framework calculates a Quality Score based on several factors, including FAIRness, license violation, maintainability, and documentation. It also generates an SBOM for the given repository. In addition, FAIRSECO provides insights into the software's impact on the RS ecosystem by gathering citation data and tracking its reuse by other software. The Impact Portal generated by FAIRSECO enables RSEs to assess the success of their software and gain insights into its impact. By showing this, we hope to raise awareness that software is valuable research output.

From the FAIRSECO report, RSEs understand the areas they must focus on to make their RS available worldwide and improve its quality. This allows users to quickly find the RS they need for their purposes, which increases its popularity. When an RS gains more users, its impact on research helps RSEs receive the credit they deserve.

In the future, we plan to:

1) *Evaluate with RSEs* – As part of the design study, we plan to present the framework to RSEs and gather their

feedback for evaluation, and further development.

2) *Promote the FAIRSECO GitHub Action* – We want to encourage RSEs to use the FAIRSECO framework. We will organize events for them and present the framework at various FAIR software events.

3) *Create a Live Dashboard for FAIRSECO Impact* – The current GitHub action is a manually triggered task. We imagine that RS projects need a continuous impact dashboard, but we will evaluate this first with the intended users.

4) *Expand the functionality of the FAIRSECO tool to support other platforms* – Currently, the framework is designed to process projects hosted on GitHub, primarily through a GitHub action. However, we aim to explore the potential of extending FAIRSECO's capabilities to include repositories or tools hosted on various other platforms. This expansion will allow for a broader range of applications and make FAIRSECO more versatile and accessible to users across different development environments.

5) *Expand documentation score* – At present, FAIRSECO's functionality is limited to checking the presence of documentation in the repository. However, in the future, our aim is to use Natural Language Processing techniques to verify complexity and extensiveness of the documentation.

6) *Introduce a dependency score* – Currently, our code reuse score is restricted to the results obtained from SEARCH-SECO. To enhance the effectiveness of the code reuse analysis, we intend to also incorporate a dependency score by detecting which other libraries and tools reuse a particular RS. This additional score will strengthen the outcomes of the code reuse analysis significantly.

7) *Score Selection and Weighting* – There are different stakeholders in the development of a research software project, such as the scientists who need it, the funding agency, and the RSEs. Each of these is interested in different scores; the scientists want to know who uses the software in their network, the funding agency wants to know whether the software is developed so it does not suffer too much from so-called software rot, and. the RSEs wish to know how many open bugs there are and their severity. For this reason different scores are required to provide insight into the different qualities of a research software project. Furthermore, it could be that composite scores are needed to rapidly provide insight into the software's qualities.

8) *Expand the notion of impact* – Ideally, a term *impact* would mean how the RS has influenced and advanced research that it was used in [33]. However, this is difficult to measure with the data available for RS. Most of the research uses quantitative impact metrics from bibliometric analysis, namely, the number of citations and code reuse (which applies to the software only). We have included these and a Quality Score as yet another factor influencing impact. In the future, we want to consider a broad notion of *impact* that includes both academic and socio-economical aspects, such as the number of grants and information regarding people hired to maintain the RS.

By extending FAIRSECO with these features, we strive to guide researchers and RSEs to adapt high quality standards for their RS, and provide more accurate insight into (potential) impact of their RS.

ARTICLE REFERENCES

[1] N. P. Chue Hong, D. S. Katz, M. Barker, A.-L. Lamprecht, C. Martinez, F. E. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, T. Honeyman, A. Struck, A. Lee, A. Loewe, B. van Werkhoven, C. Jones, D. Garijo, E. Plomp, F. Genova, H. Shanahan, J. Leng, M. Hellström, M. Sandström, M. Sinha, M. Kuzak, P. Herterich, Q. Zhang, S. Islam, S.-A. Sansone, T. Pollard, U. D. Atmojo, A. Williams, A. Czerniak, A. Niehues, A. C. Fouilloux, B. Desinghu, C. Goble, C. Richard, C. Gray, C. Erdmann, D. Nüst, D. Tartarini, E. Ranguelova, H. Anzt, I. Todorov, J. McNally, J. Moldon, J. Burnett, J. Garrido-Sánchez, K. Belhajjame, L. Sesink, L. Hwang, M. R. Tovani-Palone, M. D. Wilkinson, M. Servillat, M. Liffers, M. Fox, N. Miljković, N. Lynch, P. Martinez Lavanchy, S. Gesing, S. Stevens, S. Martinez Cuesta, S. Peroni, S. Soiland-Reyes, T. Bakker, T. Rabemanantsoa, V. Sochat, Y. Yehudi, and R. F. WG. (2022) FAIR Principles for Research Software (FAIR4RS Principles). Version: 1.0. [Online]. Available: https://doi.org/10.15497/RDA00068

[2] M. Barker, N. P. Chue Hong, D. S. Katz, M. Leggott, A. Treloar, J. van Eijnatten, and S. Aragon, "Research software is essential for research data, so how should governments respond?" Dec. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.5762703

[3] J. Carver, N. Weber, K. Ram, S. Gesing, and D. Katz, "A survey of the state of the practice for research software in the united states," *PeerJ Comput. Sci.*, vol. 8, p. e963, 05 2022. doi: 10.7717/peerj-cs.963

[4] "What is a Research Software Engineer?" 2017, accessed: 2022-04-15. [Online]. Available: https://nl-rse.org/posts/2017-06-13-what-is-rse

[5] A.-M. Istrate, D. Li, D. Taraborelli, M. Torkar, B. Veytsman, and I. Williams, "A large dataset of software mentions in the biomedical literature," *arXiv preprint arXiv:2209.00693*, 2022.

[6] M. Barker, N. P. Chue Hong, D. S. Katz, A.-L. Lamprecht, C. Martinez-Ortiz, F. Psomopoulos, J. Harrow, L. J. Castro, M. Gruenpeter, P. A. Martinez, and T. Honeyman, "Introducing the fair principles for research software," *Scientific Data*, vol. 9, no. 622, 2022. doi: 10.1038/s41597-022-01710-x

[7] D. S. Katz, M. Gruenpeter, and T. Honeyman, "Taking a fresh look at fair for research software," *Patterns*, vol. 2, no. 3, p. 100222, 2021.

[8] W. van Hage, J. Maassen, and R. van Nieuwpoort, "Lightning talk: Software Impact Measurement at the Netherlands eScience Center," *Proceedings of the Fourth Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE4)*, vol. 1686, 2016.

[9] T. Gomez-Diaz and T. Recio, "On the evaluation of research software: the CDUR procedure," *F1000Research*, vol. 8, p. 1353, nov 2019. doi: 10.12688/f1000research.19994.2. [Online]. Available: https://doi.org/10.12688/f1000research.19994.2

[10] P. Alliez, R. D. Cosmo, B. Guedj, A. Girault, M.-S. Hacid, A. Legrand, and N. Rougier, "Attributing and referencing (research) software: Best practices and outlook from inria," *Comput. Sci. Eng.*, vol. 22, no. 1, pp. 39–52, 2020. doi: 10.1109/MCSE.2019.2949413. [Online]. Available: https://ieeexplore.ieee.org/document/8887228/

[11] A.-M. Istrate, B. Veytsman, D. Li, D. Taraborelli, and I. Williams. (2022) New data reveals the hidden impact of open source in science. [Online]. Available: https://medium.com/czi-technology/new-data-reveals-the-hidden-impact-of-open-source-in-science-11cc4a16fea2

[12] S. Jansen, E. Baninemeh, and S. Farshidi, "Fairseco: An infrastructure for measuring impact of research software (short paper)," in *BElgian-NEtherlands software eVOLution symposium*, vol. 3245. CEUR Workshop Proceedings, 2022.

[13] S. Jansen, S. Farshidi, G. Gousios, J. Visser, T. van der Storm, and M. Bruntink, "Searchseco: A worldwide index of the open source software ecosystem." in *Proceedings of the 19th Belgium-Netherlands Software Evolution Workshop (BENEVOL 2020)*, ser. CEUR Workshop Proceedings, vol. 2912. CEUR-WS.org, 2020.

[14] S. Jansen, E. Handoyo, and C. Alves, "Scientists' needs in modelling software ecosystems," in *Proceedings of the 2015 european conference on software architecture workshops*, 2015, pp. 1–6.

[15] H. Borges and M. T. Valente, "What's in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018.

[16] D. Singh Chawla, "The unsung heroes of scientific software," *Nature*, vol. 529, p. 115–116, 2016. doi: 10.1038/529115a

[17] P. Boldi, A. Pietri, S. Vigna, and S. Zacchiroli, "Ultra-large-scale repository analysis via graph compression," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 184–194.

[18] S. Dueñas, V. Cosentino, J. M. Gonzalez-Barahona, A. d. C. San Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, and A. P. García-Plaza, "Grimoirelab: A toolset for software development analytics," *PeerJ Computer Science*, vol. 7, p. e601, 2021.

[19] P. Orviz, J. Gomes, S. Bernardo, D. Naranjo, M. David, and EOSC-SYNERGY, "EOSC-SYNERGY EU DELIVERABLE: D3.4 Final release of the SQAaaS," 2022. [Online]. Available: http://hdl.handle.net/10261/274895

[20] A. Devaraju and R. Huber, "An automated solution for measuring the progress toward FAIR research data," *Patterns*, vol. 2, no. 11, 2021. doi: 10.1016/j.patter.2021.100370. [Online]. Available: https://doi.org/10.1016/j.patter.2021.100370

[21] J. Priem, H. Piwowar, and R. Orr, "Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts," 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.01833

[22] C. Martinez-Ortiz, M. Kuzak, J. H. Spaaks, J. Maassen, and T. Bakker, "Five recommendations for "FAIR software"," Dec. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4310217

[23] C. Martinez-Ortiz and J. H. Spaaks. (2021) Fair Software Recommendations. [Online]. Available: https://fair-software.nl/home/

[24] A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. Martin Del Pico, V. Dominguez Del Angel, S. Van De Sandt, J. Ison, P. A. Martinez *et al.*, "Towards fair principles for research software," *Data Science*, vol. 3, no. 1, pp. 37–59, 2020.

[25] A. M. S. Laurent, *Understanding Open Source and Free Software Licensing: guide to navigating licensing issues in existing & new software*. O'Reilly Media, Inc., 2004. ISBN 9780596553951

[26] G. R. Gangadharan, V. D'Andrea, S. De Paoli, and M. Weiss, "Managing license compliance in free and open source software development," *Information Systems Frontiers*, vol. 14, pp. 143–154, 05 2012. doi: 10.1007/s10796-009-9180-1

[27] N. Schneidewind, "The state of software maintenance," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 3, pp. 303–310, 1987. doi: 10.1109/TSE.1987.233161

[28] R. Malhotra and A. Chug, "Software maintainability: Systematic literature review and current trends," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 08, pp. 1221–1253, 2016.

[29] P. Avgeriou, P. Kruchten, I. Ozkaya, and S. Carolyn, "Managing technical debt in software engineering," in *Report from Dagstuhl Seminar 16162*, ser. Dagstuhl reports, vol. 6, no. 4, 2016. doi: 10.4230/DagRep.6.4.110

[30] C. Martinez-Ortiz, P. Martinez Lavanchy, L. Sesink, B. G. Olivier, J. Meakin, M. de Jong, and M. Cruz, "Practical guide to software management plans," Jan. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7589725

[31] S. Arsyad, M. Zaim, and D. Susyla, "Review and citation style in research article introductions: a comparative study between national and international english-medium journals in medical sciences," *Discourse and Interaction*, vol. 11, no. 1, pp. 28–51, 2018.

[32] R. Haghnazar Koochaksaraei, F. Gadelha Guimarães, B. Hamidzadeh, and S. Hashemkhani Zolfani, "Visualization method for decision-making: a case study in bibliometric analysis," *Mathematics*, vol. 9, no. 9, p. 940, 2021.

[33] T. Penfield, M. J. Baker, R. Scoble, and M. C. Wykes, "Assessment, evaluations, and definitions of research impact: A review," *Research Evaluation*, vol. 23, no. 1, pp. 21–32, 10 2013. doi: 10.1093/reseval/rvt021. [Online]. Available: https://doi.org/10.1093/reseval/rvt021

SOFTWARE REFERENCES

[34] J. H. Spaaks, S. Verhoeven, E. Tjong Kim Sang, F. Diblen, C. Martinez-Ortiz, E. Etuk, M. Kuzak, B. van Werkhoven, A. Soares Siqueira, S. Saladi, and A. Holding, "**HOWFAIRIS**," 9 2022, version: 0.14.2. [Online]. Available: https://github.com/fair-software/howfairis

[35] S. Druskat, J. H. Spaaks, N. Chue Hong, R. Haines, J. Baker, S. Bliven, E. Willighagen, D. Pérez-Suárez, and O. Konovalov, "**Citation File Format**," Aug. 2021.

[36] S. Verhoeven, F. Diblen, J. H. Spaaks, and E. Tjong Kim Sang, "**tortellini**," 6 2021, version: v3. [Online]. Available: https://github.com/tortellini-tools/action

[37] A. Aydin, B. Hageman, B. Lankhorst, J. Hendriksen, Q. Donkers, R. Schouten, T. Bolhuis, V. Bykova, and The FAIRSECO contributors, "**FAIRSECO**," 2 2023, version : 1.0.0. [Online]. Available: https://github.com/QDUNI/FairSECO

[38] S. Verhoeven, F. Diblen, J. H. Spaaks, and E. Tjong Kim Sang, "**fairtally**," 3 2021. doi: 10.5281/zenodo.4590883 Version: 0.1.0. [Online]. Available: https://github.com/fair-software/fairtally

[39] A. Nesbitt, T. Zaharia, J. Katz, and M. Young, "**Libraries.io**." [Online]. Available: https://github.com/librariesio

[40] J. H. Spaaks, T. Klaver, S. Verhoeven, F. Diblen, J. Maassen, E. Tjong Kim Sang, P. Pawar, C. Meijer, L. Ridder, L. Kulik, T. Bakker, V. van Hees, L. Bogaardt, A. Mendrik, B. van Es, J. Attema, W. van Hage, E. Ranguelova, R. van Nieuwpoort, R. Gey, and H. Zach, "**Research Software Directory**," 2020. doi: 10.5281/zenodo.1154130 Version: 3.0.1. [Online]. Available: https://github.com/research-software-directory/research-software-directory

[41] D. M. Naranjo Delgado, D. Arce Grilo, P. Orviz Fernández, and S. Bernardo, "**SQAaaS Web**," https://github.com/EOSC-synergy/sqaaas-web.

[42] R. W. Field, "**Semantic Scholar**." [Online]. Available: https://www.semanticscholar.org/product/api

[43] A. Malladi, A. Kovalyov, K. Sigmund, E. Ruiz, and J. Wen, "**SBOM**." [Online]. Available: https://github.com/microsoft/sbom-tool

[44] D. van Kuppevelt, C. Meijer, F. Huber, V. van Hees, B. Solino Fernandez, P. Bos, J. Spaaks, M. Kuzak, J. Hidding, A. van der Ploeg, M. Lüken, and O. Lyashevska, "**mcfly**: deep learning for time series," Dec. 2022. doi: 10.5281/zenodo.596127 Version :4.0.0. [Online]. Available: https://github.com/NLeSC/mcfly

[45] B. van Werkhoven, "**Kernel Tuner**," 2019. doi: 10.5281/zenodo.1220113 Version: 0.3.0. [Online]. Available: https://github.com/benvanwerkhoven/kernel_tuner

[46] F. Huber, J. J. J. van der Hooft, J. H. Spaaks, F. Diblen, S. Verhoeven, C. Geng, C. Meijer, S. Rogers, A. Belloum, H. Spreeuw, N. de Jonge, and M. Skoryk, "**spec2vec**." [Online]. Available: https://github.com/iomega/spec2vec

[47] B. Andela, B. Broetz, L. de Mora, N. Drost, V. Eyring, N. Koldunov, A. Lauer, B. Mueller, V. Predoi, M. Righi, M. Schlund, J. Vegas-Regidor, K. Zimmermann, K. Adeniyi, E. Arnone, O. Bellprat, P. Berg, L. Bock, L.-P. Caron, N. Carvalhais, I. Cionni, N. Cortesi, S. Corti, B. Crezee, E. L. Davin, P. Davini, C. Deser, F. Diblen, D. Docquier, L. Dreyer, C. Ehbrecht, P. Earnshaw, B. Gier, N. Gonzalez-Reviriego, P. Goodman, S. Hagemann, J. von Hardenberg, B. Hassler, A. Hunter, C. Kadow, S. Kindermann, S. Koirala, L. Lledó, Q. Lejeune, V. Lembo, B. Little, S. Loosveldt-Tomas, R. Lorenz, T. Lovato, V. Lucarini, F. Massonnet, C. W. Mohr, E. Moreno-Chamarro, P. Amarjiit, N. Pérez-Zanón, A. Phillips, J. Russell, M. Sandstad, A. Sellar, D. Senftleben, F. Serva, J. Sillmann, T. Stacke, R. Swaminathan, V. Torralba, K. Weigel, C. Roberts, P. Kalverla, S. Alidoost, S. Verhoeven, B. Vreede, S. Smeets, A. Soares Siqueira, and R. Kazeroni, "**ESMValTool**," 3 2023, version: v2.8.0. [Online]. Available: https://doi.org/10.5281/zenodo.7778410

[48] J. Liu, "**LlamaIndex**," 11 2022. doi: 10.5281/zenodo.1234. [Online]. Available: https://github.com/jerryjliu/gpt_index

[49] J. H. Spaaks, T. Klaver, S. Verhoeven, S. Druskat, and W. Leoncio, Netto, "**cffconvert**," 9 2021, version: 2.0.0. [Online]. Available: https://github.com/citation-file-format/cff-converter-python