FAIR Jupyter: A Knowledge Graph Approach to Semantic Sharing and Granular Exploration of a Computational Notebook Reproducibility Dataset

Sheeba Samuel[1]
Distributed and Self-organizing Systems, Chemnitz University of Technology, Chemnitz, Germany

Daniel Mietchen[1]
FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany
Institute for Globally Distributed Open Research and Education (IGDORE)

Abstract

The way in which data are shared can affect their utility and reusability. Here, we demonstrate how data that we had previously shared in bulk can be mobilized further through a knowledge graph that allows for much more granular exploration and interrogation. The original dataset is about the computational reproducibility of GitHub-hosted Jupyter notebooks associated with biomedical publications. It contains rich metadata about the publications, associated GitHub repositories and Jupyter notebooks, and the notebooks' reproducibility. We took this dataset, converted it into semantic triples and loaded these into a triple store to create a knowledge graph – FAIR Jupyter – that we made accessible via a web service. This enables granular data exploration and analysis through queries that can be tailored to specific use cases. Such queries may provide details about any of the variables from the original dataset, highlight relationships between them or combine some of the graph's content with materials from corresponding external resources. We provide a collection of example queries addressing a range of use cases in research and education. We also outline how sets of such queries can be used to profile specific content types, either individually or by class. We conclude by discussing how such a semantically enhanced sharing of complex datasets can both enhance their FAIRness – i.e., their findability, accessibility, interoperability, and reusability – and help identify and communicate best practices, particularly with regards to data quality, standardization, automation and reproducibility.

2012 ACM Subject Classification Information systems → Entity relationship models; Information systems → Information extraction

Keywords and phrases Knowledge Graph, Computational reproducibility, Jupyter notebooks, FAIR data, PubMed Central, GitHub, Python, SPARQL

Digital Object Identifier 10.4230/TGDK.2.2.4

Category Resource Paper

Related Version *Full Version*: `https://doi.org/10.48550/arXiv.2404.12935`

Supplementary Material see also Supplementary Material Statement in Section 7
Software (Source Code): `https://doi.org/10.5281/zenodo.14197755` [69]
Service (Website): `https://w3id.org/fairjupyter` [68]

Funding *Sheeba Samuel*: German Research Foundation (DFG), TRR-386, project number 514664767
Daniel Mietchen: MaRDI: DFG 460135501, BASE4NFDI/KGI4NFDI: DFG 521453681

Acknowledgements We thank the providers of infrastructure, data, and code that we used in this study. These include the PubMed Central repository at the U.S. National Center for Biotechnology Information and the Ara Cluster at the University of Jena as well as the Jupyter, Python and Conda communities and their respective dependencies. The authors gratefully acknowledge the computing

[1] Corresponding author. Both authors contributed equally to this work.

time made available to them on the high-performance computer at the NHR Center of TU Dresden. This center is jointly supported by the Federal Ministry of Education and Research and the state governments participating in the NHR (`www.nhr-verein.de/unsere-partner`). Special thanks go to JupyterCon, which provided the nucleus for our collaboration. We also thank Ramy-Badr Ahmed and Moritz Schubotz for help with registering the GitHub repositories from our corpus in the Software Heritage archive.

Received 2024-07-01 **Accepted** 2024-11-04 **Published** 2024-12-18

Editors Aidan Hogan, Ian Horrocks, Andreas Hotho, Lalana Kagal, and Uli Sattler

Special Issue Resources for Graph Data and Knowledge

1 Introduction

In an age where research findings shape policy decisions and impact our understanding of the world, ensuring the reproducibility of scientific work is paramount. Jupyter notebooks [43] have revolutionized the way researchers share code, results, and documentation, all within an interactive environment, promising to make science more transparent and reproducible [30]. In research contexts, Jupyter notebooks often coexist with other software and various resources such as data, instruments, and mathematical models, all of which may affect scientific reproducibility. The evolving Jupyter ecosystem and the growing popularity of code sharing platforms like GitLab, Gitee, or Codeberg in parallel with GitHub require systematic approaches in future assessments of Jupyter reproducibility.

The FAIR principles – Findable, Accessible, Interoperable, and Reusable – play a crucial role in promoting effective data sharing practices across scientific disciplines, thereby enhancing reproducibility [80]. Data sharing has many facets, including the nature of the data, the purpose of the sharing, reuse considerations as well as various practicalities like the choice of file formats, metadata standards, licensing and location for the data to be shared. Here, we look into some of the practical and reuse aspects by mobilizing a previously shared reproducibility dataset in a more user-friendly fashion. That previous dataset arose from a study [71] of the computational reproducibility of Jupyter notebooks associated with biomedical publications. It is already publicly available [70] as a 1.5GB SQLite database contained within a ZIP archive of 415.6 MB (compressed) but in order to be explored, it needs to be unzipped, loaded into a SQLite server and queried via SQL. While these steps are routine for many, they nonetheless present a technical hurdle that stands in the way of broader use of the data, both in research and educational contexts.

For instance, imagine an instructor of a programming course for wet lab researchers who wants to present to her students some real-world Jupyter notebooks from their respective research field that use a specific Python module and are either fully reproducible or exhibit a given type of error. Wouldn't it be nice – and quicker than via the route outlined above – if she could get her students to run such queries directly in their browser, with no need to install anything on their system? Enabling students and instructors to do this is what we are aiming at. Likewise, reproducibility researchers can explore what aspects of notebooks, repositories, journals or papers correlate with reproducibility-related variables, editors can become aware of common issues in notebooks associated with their publications venue, while maintainers of software packages, ontologies or Jupyter-based services can explore the use of their resources in a reproducibility context.

One way to build a webservice addressing such use cases would be to use a web framework like Flask or Django in combination with a library like sqlite3 [10] to interact with the database. However, this approach complicates semantic integration with other resources, and so we chose instead to leverage semantic web standards [53] and build a demonstrator for converting a dataset into a knowledge graph (KG). For people interested in granular exploration of datasets by way of knowledge graphs, it provides both a blueprint on how to get started and a working prototype to check against.

In this paper, we thus introduce **FAIR Jupyter**, a KG created from Jupyter notebooks hosted on GitHub and linked to biomedical publications sourced from PubMedCentral [59]. This resource is aimed at the intersection between KGs, reproducibility and Jupyter/Python: readers can engage with these individual areas according to their expertise, optionally in a way that is assisted by knowledge in the other areas. Here, we outline our two primary contributions: (i) The **FAIR Jupyter Ontology** developed using the NeOn methodology [76] reuses and extends the state-of-the-art ontologies to describe metadata related to the reproducibility of Jupyter notebooks, GitHub repositories, publications, and journals. (ii) The **FAIR Jupyter Knowledge Graph**, a KG containing 190 million triples about GitHub-hosted Jupyter notebooks extracted from PubMedCentral, and their reproducibility. In addition to metadata on repositories, journals, publications, and authors, the KG also includes fine-grained information on atomic elements of notebooks including cells, input, output, data and code dependencies, modules, libraries, styling, executions, execution order, cell features, and errors.

This emerging resource available as a KG identifies various issues, including imprecise statements of requirements such as data and code dependencies, the use of custom software libraries, inadequate documentation, the use of hard-coded paths, non-descriptive filenames, non-open data as well as the nature of default settings or policies used by deployment frameworks and infrastructures. We provide real-world examples highlighting specific problems, serving as a foundation for addressing such problems in order to improve computational reproducibility. Besides using the KG to showcase Jupyter-related information, we use Jupyter notebooks to showcase information about the KG. Additionally, we present SPARQL queries designed to achieve several objectives: (1) making it straightforward to reproduce results from [71] (this was already possible with the original data, but required more technical expertise and user-side infrastructure); (2) making it easier to filter the dataset for subgraphs of interest, e.g. things not covered in detail in [71], such as notebooks associated with a particular journal or written in languages other than Python; (3) combining information from this KG with information from external sources through federated queries.

The platform does not only target authors of research software like Jupyter notebooks but also addresses what other stakeholders in the research ecosystems can do to enhance computational aspects of reproducibility, such as providers of research infrastructures on which such computations are run, as well as reviewers, editors, and publishers of manuscripts reporting on computational aspects of research. It can be used by educators to provide practical tips to avoid common pitfalls and improve the reproducibility of computational analyses, particularly when conducted and shared through notebook environments like Jupyter.

Our work represents a significant milestone as the first systematic and large-scale effort to crawl, integrate, and semantically publish repositories of research-related Jupyter notebooks as a KG and to enrich that KG with reproducibility information pertaining to these notebooks. The current setup paves the way from the original one-off snapshots of Jupyter reproducibility to a future service that can provide a routinely updated dashboard-like overview of the Jupyter reproducibility landscape in biomedical research, while at the same time stimulating exploration and education around the role of knowledge graphs in this space.

2 Motivation and Use Cases

The need for reproducibility in computational research has become increasingly critical, particularly in fields that rely heavily on data science and computational methods. Jupyter notebooks are widely used to document and share these computational workflows, but ensuring their reproducibility poses significant challenges [55]. By creating a structured resource that integrates Jupyter notebooks with their associated publications and repositories, we aim to provide a comprehensive platform for assessing and enhancing computational reproducibility and contributes to scientific reproducibility.

One of the key motivations for converting the existing dataset into a KG is to enable more granular and flexible querying of reproducibility information. KGs offer a dynamic and interconnected way to represent data, enabling users to explore and analyze relationships between different entities, in this context, journals, articles, authors, repositories, and specific notebook cells, as highlighted in Table 5. For example, users can easily query for specific error types across notebooks from different journals or research fields. With federated queries, a KG allows users to combine data from multiple sources, such as Wikidata, to enrich the analysis with additional contextual information about papers, authors, affiliations, and software packages. With KGs, researchers find relevant data and reproducibility information more efficiently, enhancing the discoverability of datasets by exposing relationships and metadata that might be buried in traditional databases. The motivation for developing this resource is also based on discussions from our community engagements [6] for reuse in a more user-friendly way.

Through a KG, researchers can query aggregated data across journals, articles, and repositories to analyze trends and patterns in computational reproducibility. They can investigate GitHub repositories to understand the structure and dependencies of computational workflows. Detailed queries at the notebook level allow for the examination of individual Jupyter notebooks, assessing their reproducibility and computational methods. Queries at the author level help study patterns of computational practices and reproducibility across different contributors. Granular queries at the level of individual cells or markdowns within notebooks provide insights into the execution and documentation of computational steps. Researchers can target specific requirements and dependencies stated within notebooks, highlighting areas critical for reproducibility. Subject-level queries enable the exploration of computational reproducibility trends within specific research fields or disciplines. Information from different entity types can also be combined in various fashions, e.g. to highlight exceptions that are common for notebooks associated with publications in a given field or journal. Additionally, queries can focus on the stylistic aspects of notebooks, examining conventions and best practices for documentation and presentation. Table 5 provides motivating examples of queries that users can explore. These use cases demonstrate the versatility of the resource in supporting varied research needs and enhancing computational reproducibility across different dimensions of scholarly communication and practice.

3 Related Works

This section explores the current state-of-the-art approaches in computational reproducibility, along with the ontologies and KGs that have been developed to support these efforts.

Computational Reproducibility

Multiple studies have explored the reproducibility of computational research. For example, Gruning et al. [33] examined the specifics of computational reproducibility in the life sciences. Nust et al. [52] investigated the use of Docker, a containerization tool, in reproducibility contexts. Trisovic et al. [77] focused on the reproducibility of R scripts archived in an institutional repository.

Several studies have been conducted in recent years to explore the reproducibility of Jupyter notebooks [60, 55, 74, 79, 81]. Rule et al. [61] examined one million notebooks available on GitHub, exploring repositories, languages, packages, notebook length, and execution order, with a focus on the structure and formatting of computational notebooks. This study resulted in the proposal of ten best practices for writing and sharing computational analyses in Jupyter notebooks [60].

Another study [56] focused on the reproducibility of 1.4 million notebooks collected from GitHub, providing an extensive analysis of the factors impacting reproducibility in Jupyter notebooks. With respect to the use of Jupyter notebooks in research contexts, Chattopadhyay et al. [19] reported on a survey conducted among 156 data scientists, highlighting the challenges they face when working with notebooks, while Schröder et al. [74] manually examined the reproducibility of Jupyter notebooks linked to five publications from PubMed Central .

The datasets generated from these studies, although not directly linked to any specific publications, face significant challenges. These datasets are often not easily queryable or reusable in a user-friendly manner. Additionally, they cannot be seamlessly linked or integrated using federated queries with other valuable sources such as Wikidata or DBpedia. This lack of interoperability and accessibility hinders the potential for broader analysis and insights, limiting their usability and impact within the research community.

Ontologies

Semantic web technologies and ontologies play a crucial role in enhancing computational reproducibility. Ontologies, in particular, offer a standardized vocabulary and relationships that facilitate the consistent annotation of computational research, making it easier to understand and reproduce experiments [28, 29, 45, 67]. For instance, the use of ontologies such as PROV-O [45] and P-Plan [28] allows researchers to describe the provenance of data and the specifics of computational processes, ensuring that every step of analysis can be traced. Here, we focus on the ontologies that represent computational workflows. The existence of numerous well-established ontologies points to the maturity of the scientific workflow domain [28, 45]. A significant number of ontologies have also been developed in the computer science and artificial intelligence domain to describe computational processes [25, 63]. The REPRODUCE-ME ontology [64, 65] is the pioneering effort to describe the provenance of Jupyter notebooks by extending the PROV-O [45] and P-Plan [28] ontologies. As this work touches upon different areas like publication [54], journals, repositories [3], and notebooks [67], it is important that well-established ontologies are reused to describe the domain of scholarly publications [21] and computational reproducibility.

Knowledge Graphs

While we are not aware of KGs about Jupyter notebooks or GitHub repositories, there have been a number of related efforts. These include the creation of KGs about FAIR computational workflows [29], PubMed [83], scientific software [40] and artificial intelligence tasks and benchmarks [15], along with tools for creating schemas and KGs from data [37]. Of particular relevance here is the Open Research Knowledge Graph (ORKG) [14] which provides a framework to represent, curate, and discover scholarly knowledge in a structured manner. It has also shown potential in assisting with reproducibility [39, 38]. To address scientific reproducibility in biomedicine, Liu et al. [48] developed ProvCaRe, a semantic provenance resource that was aggregating information extracted using NLP from the full text of 1.6 million biomedical articles. The repository (now defunct) required users to log in and contained 166 million provenance triples focusing on Study Method, Study Data, and Study Tool (including software, though computational reproducibility was not assessed). Another study [62] presents a semantic provenance graph from 75 sleep medicine articles, mapping provenance information to PROV-O [45] for querying reproducibility-related information, albeit not in a Jupyter context. The Microsoft Academic Knowledge Graph [26] was a large-scale RDF dataset with information on scholarly publications, authors, institutions, journals, and subject areas. It has been used to analyze repositories and their associated papers based on metrics like stars, forks, and the number of contributors [27]. Information about GitHub repositories and GitHub contributors has been combined with information from the Wikidata KG to highlight gender-specific contribution patterns in open-source software projects [46]. An initial attempt has also been made to create a KG of Git repositories using a prototype tool called GitGraph. This tool extracts metadata from repositories, including commits and files, and constructs a KG. Graph4Code [12] is another relevant effort that is designed to generate KGs from code, supporting applications like program search, code understanding, bug detection, and automation. It complements our efforts, as its focus is on representing the structure of Python scripts and the flow of data through a script's components, rather than computational reproducibility.

These efforts collectively contribute to a better understanding and improvement of reproducibility in computational research, highlighting the importance of structured metadata, provenance, and advanced tools for maintaining the integrity and reliability of scientific workflows. To our knowledge, no prior work has undertaken a systematic effort to describe and semantically publish large-scale reproducibility analyses of Jupyter notebooks from research publications as a KG.

4 Methods

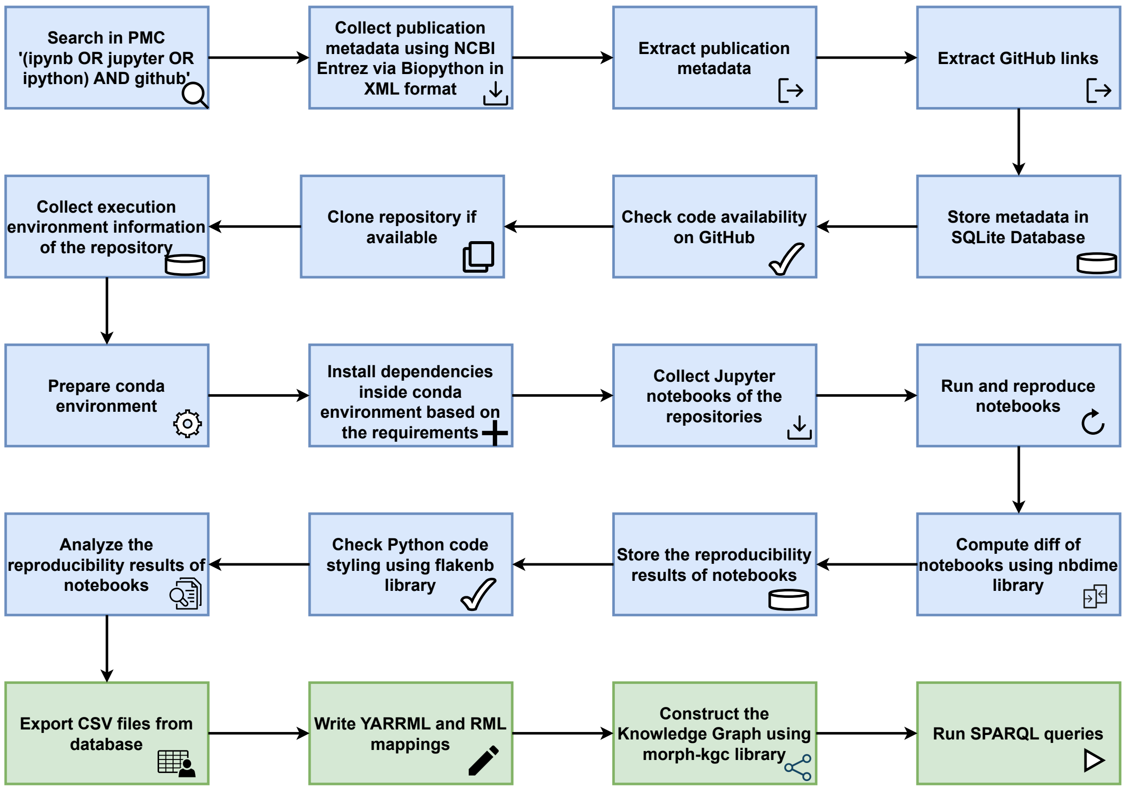
Search in PMC '(ipynb OR jupyter OR ipython) AND github'
Collect publication metadata using NCBI Entrez via Biopython in XML format
Extract publication metadata
Extract GitHub links
Store metadata in SQLite Database
Check code availability on GitHub
Clone repository if available
Collect execution environment information of the repository
Prepare conda environment
Install dependencies inside conda environment based on the requirements
Collect Jupyter notebooks of the repositories
Run and reproduce notebooks
Compute diff of notebooks using nbdime library
Store the reproducibility results of notebooks
Check Python code styling using flakenb library
Analyze the reproducibility results of notebooks
Export CSV files from database
Write YARRML and RML mappings
Construct the Knowledge Graph using morph-kgc library
Run SPARQL queries

Figure 1 Workflow overview. The blue workflow was used to construct the original dataset [70] and is described in [71], while the subsequent KG construction workflow in green represents the current study.

Our workflow has two main components, as shown in Figure 1. The first is the generation of the Jupyter notebook reproducibility dataset. The second is the conversion of this dataset into the FAIR Jupyter KG, which forms the focus of the present study.

4.1 Computational Reproducibility Dataset Generation

This section represents a summary of the methodology used in [71]. Briefly, we queried PubMed Central (PMC, cf. [59]) for publications mentioning GitHub alongside keywords such as "Jupyter", "ipynb" (the file extension for Jupyter notebooks), or "IPython" (a predecessor to Jupyter).[2] Utilizing the primary PMC IDs obtained this way, we then retrieved publication records in XML format using the *efetch* function and collected publication metadata from PMC using NCBI Entrez utilities via Biopython [22].

Next, we processed the XML data retrieved from PMC by storing it in an SQLite database. Our database encompassed details regarding journals and articles, populating it with metadata including ISSN (International Standard Serial Number), journal and article titles, PubMed IDs, PMC IDs, DOIs, subjects, submission, acceptance, and publication dates, licensing information, copyright statements, keywords, and GitHub repository references mentioned in the publication. Additionally, we extracted associated Medical Subject Headings (MeSH terms) [8] for each article. These terms, assigned during indexing in the PubMed database, are hierarchical. We obtained the top-level MeSH term by querying the MeSH RDF API through SPARQL queries to the SPARQL endpoint [9]. This aggregation resulted in 108 top-level MeSH terms in our dataset, serving as proxies for the subject areas of the articles.

We extracted the GitHub repositories mentioned in each article, including the abstract, body, data availability statement, and supplementary sections.[3] After this preprocessing, we associated each article with the GitHub repositories extracted from it as well as with the journal in which the article had been published, and we gathered author information in a separate database table, including first and last names, ORCID, and email addresses.

We verified the availability of GitHub repositories mentioned in the articles and, if existing, cloned them, based on the main branch, and gathered repository details including creation, update, and push dates, releases, issues, license details, etc. using the GitHub REST API [5]. Additionally, we extracted details for each notebook provided in the repository, such as name, nbformat, kernel, language, cell types, and maximum execution count, and extracted source code and output from each cell using Python Abstract Syntax Tree (AST) for further analysis.

After the notebook collection, we gathered execution environment details by examining dependency declarations in repository files like requirements.txt, setup.py, and pipfile. After collecting the necessary Python notebook execution information, we prepared a conda environment based on the declared Python version, installing dependencies listed in files such as requirements.txt, setup.py, and pipfile. For repositories lacking specified dependencies, the pipeline executed notebooks by installing all Anaconda dependencies, leveraging Anaconda's comprehensive data science package suite. We also conducted Python code styling checks using the flakenb [4] library, which enforces code style guidelines outlined in PEP 8, to collect all detected errors, obtaining information on the error code and description.

We executed our pipeline on 27th March 2023, and it ran until 9 May 2023, for a total of 43 days. The code has been adapted from [55, 66]. We utilize this method to reproduce Jupyter notebooks from GitHub repositories, as outlined in [55]. Additionally, we leverage ReproduceMeGit [66], which uses the nbdime library [57] to compare execution results with the original results. This forms the foundation of our code for the reproducibility study.

[2] We used the search query "(ipynb OR jupyter OR ipython) AND github".
[3] Since the GitHub repositories had been stated in a number of different formats, we harmonized them to "https://github.com/username/repositoryname".

4.2 FAIR Jupyter Ontology and KG Construction

Competency Questions (CQs)

The requirements for ontology construction are driven by the requirements used for generating the initial dataset and the CQs are based on the research questions that arose from the initial pipeline. We used the NeOn methodology [76] for constructing the FAIR Jupyter ontology. We present the Ontology Requirement Specification Document (ORSD) which outlines the purpose, scope, implementation language, intended end-users and uses of the ontology, and the set of requirements the ontology should fulfill, presented in the form of competency questions. These competency questions are organized into eight categories, addressing the domain knowledge that needs to be represented, as detailed in Table 5 and 6.

Data modeling

In this section, we provide a brief description of the ontology model used in the construction of the FAIR Jupyter KG. Overall, it contains 22 classes, as outlined in Figure 2. They are centred around notebooks, notebook cells, repositories and articles, each of which are linked to several other classes. We reused the following ontologies for describing these entities: PROV-O [45], REPRODUCE-ME [64], P-Plan [28], PAV [21], DOAP [3] and FaBiO [54].

Building on PROV and P-Plan, the REPRODUCE-ME ontology captures provenance information for individual Jupyter Notebook cells [64], in addition to the end-to-end scientific experiment with real-life entities like instruments and specimens, as well as human activities such as lab protocols and screening [67]. Hence, we used and extended this model to construct the FAIR Jupyter KG.

We reused `fabio:Article` to link the publications in our dataset and `fabio:Journal` to represent the journal where the article was published. FaBiO is an ontology that helps represent information about publishable works (articles, books, etc.) and the bibliographic references that connect them [54]. A GitHub repository is represented using the class `doap:GitRepository` from the DOAP ontology [3], which is used to describe open source software projects. We reused the class `repr:Notebook` to represent the Jupyter Notebooks, which is a subclass of `p-plan:Plan` extending the class `prov:Plan`. The PROV-O [45] is a W3C recommendation providing foundation to implement provenance applications. We use two object properties of the PROV-O ontology: `prov:specializationOf` and `prov:generalizationOf` to show that the article is a specialization of a MESH term and the corresponding MESH term a generalisation of the top level MESH term in its hierarchy. To denote where a resource is retrieved from (e.g., `repr:Notebook` is `pav:retrievedFrom doap:GitRepository`), we have used the object property `pav:retrievedFrom`. PAV which builds on the PROV-O ontology, is a lightweight ontology for tracking Provenance, Authoring and Versioning and describes information about authorship, curation, and versioning of online resources [21]. Each individual cell of a computational notebook `repr:Cell` is described as a step of a `repr:Notebook` using the object property `p-plan:isStepOfPlan`. To describe the different features of notebooks and their cells, we reused the object property `p-plan:isVariableOfPlan`.

Mappings and KG construction

We used the CSV files exported from the database [70] as input for the YARRRML mapping [36]. YARRRML is a human-readable text-based format for expressing declarative rules to generate Linked Data from different data sources. Listing 1 shows an example of a YARRRML mapping used for expressing the rules for “repositories” and “notebooks”. To facilitate adaptation to a variety of

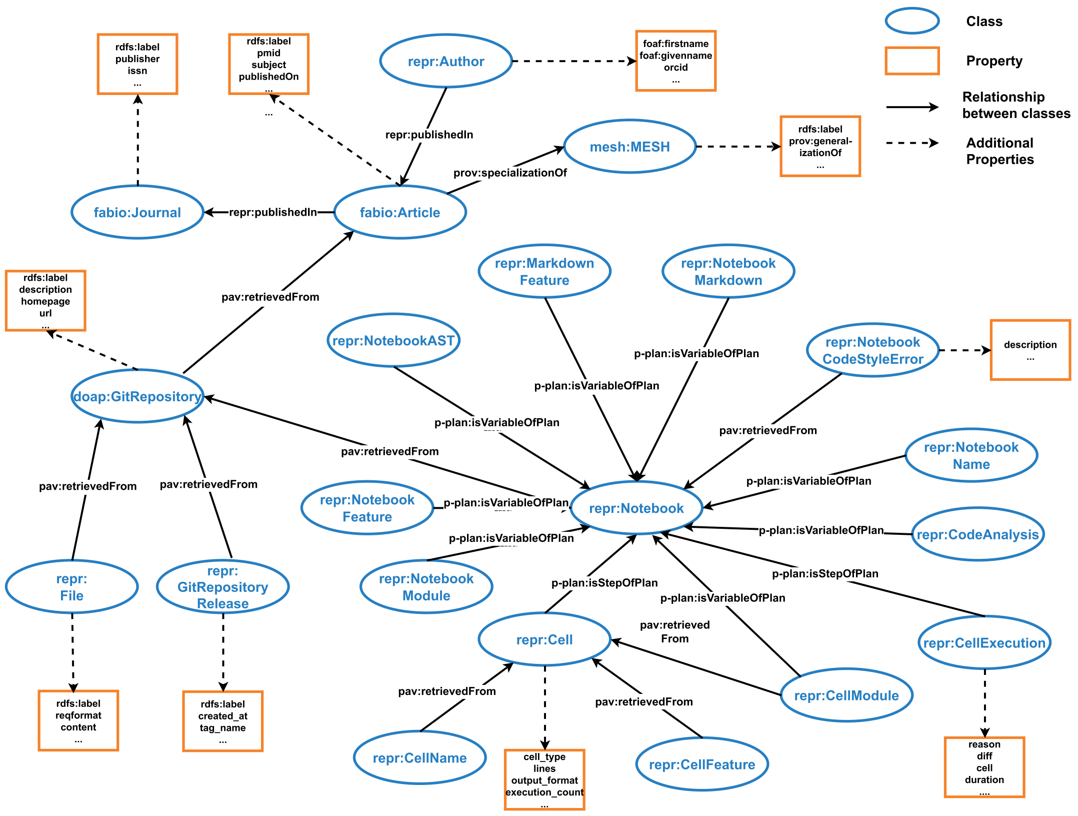

Figure 2 Partial outline of the data model used in FAIR Jupyter. Classes of entities (represented by ellipses) and the class properties (represented by orange rectangles) were inferred from the original dataset, and – along with relationships between them (arrows) – expressed in terms of relevant ontologies. Note that requirement files and repository files are both represented as `repr:File`. While some properties have not been depicted here for clarity and some classes have not been included in our endpoint for performance reasons, all the underlying files – CSV exports of the original data, the RML and YARRML mapping files used for KG construction, and the resulting RDF triples – are available via [72].

.

use cases, we created mappings for each entity types in separate files. The mappings are created for the classes of the ontology (see Figure 2). We chose the same name for the user-chosen keys as the name provided in the database tables. The key source has the value of the corresponding CSV file of the entity type. We used the namespace `https://w3id.org/reproduceme/` for generating the IRI. We reused the properties from the existing ontologies wherever possible to generate the combination of predicates and objects. For others, we added them to the REPRODUCE-ME ontology. The YARRRML mappings were written using the YARRRML editor, Matey [7]. These mappings were then converted to RML mappings [24]. The generated RML mappings were further used as input to the Morph-KGC library [13]. Morph-KGC is a tool designed to build RDF knowledge graphs from different data sources using the R2RML and RML mapping languages. For large datasets, Morph-KGC's use of mapping partitions significantly speeds up processing and reduces memory usage. For some classes (CellName, CodeAnalysis, RepositoryFile, and MarkdownFeature), their size triggered import errors, and so we excluded them from this prototype but we plan to include them in the future as we continue to develop the platform. All the triples generated were stored in N-triples format and are downloadable in bulk from [72].

Listing 1 Part of a YARRRML mapping for repositories and notebooks.

```
mappings:
repositories:
    sources:
      - [data/repositories.csv~csv]
    s: https://w3id.org/reproduceme/repository_$(id)
    po:
      - [a, doap:GitRepository]
      - [rdfs:label,$(repository)]
      - p: pav:retrievedFrom
        o:
          - mapping: article
            condition:
              function: equal
              parameters:
                - [str1, $(article_id), s]
                - [str2, $(id), o]
  notebooks:
    sources:
      - [data/notebooks.csv~csv]
    s: https://w3id.org/reproduceme/notebook_$(id)
    po:
      - [a, repr:Notebook]
      - [rdfs:label, $(name)]
      - [repr:kernel, $(kernel)]
      - [repr:language, $(language)]
      - p: pav:retrievedFrom
        o:
          - mapping: repositories
            condition:
              function: equal
              parameters:
                - [str1, $(repository_id), s]
                - [str2, $(id), o]
```

Triple store

We use Apache Jena Fuseki [18] as a triple store to query our KG. The FAIR Jupyter KG is available at `https://reproduceme.uni-jena.de/#/dataset/fairjupyter/query`.

Web interface for FAIR Jupyter: visual exploration and querying

We provide a web service at `https://reproduceme.uni-jena.de/fj` that facilitates the visual exploration and interactivity of the FAIR Jupyter ontology and knowledge graph. Through this platform, users can directly explore the KG via a browser interface written using vis.js [11] that displays all KG entities. By selecting an entity, users can view detailed attributes, access links to relevant ontologies, and examine all associated properties. The service also enables users to run their own SPARQL queries, providing a flexible environment for custom exploration. In addition, the website features a dedicated section showcasing all queries used in this paper, including those that reproduce the original results from [71], as well as additional queries – including federated ones – highlighting aspects of the data that were not discussed there. To assist users from various backgrounds, a comprehensive documentation page is provided, consolidating all resources and guidance in one place for ease of use.

5 Results

Table 1 shows some general statistics about the FAIR Jupyter KG, which consists of about 190 million triples taking up a total of about 20.6 GB in space. Taking inspiration from the Scholia frontend to Wikidata [51], we anticipate using the KG for creating profiles based on specific entity types. To this end, we created the FAIR Jupyter KG from multiple smaller graphs based on separate mappings for each entity type. We present the time it took to construct the KG, the number of mapping rules retrieved, the total triples generated for each entity and the total file size of the RDF file generated.

Table 1 General statistics of the FAIR Jupyter KG. The graphs for four entity types (CellName, CodeAnalysis, RepositoryFile, and MarkdownFeature) were omitted from the prototype implementation for performance reasons but included here, as we plan to include them in future versions of the KG.

Entity	Time (in sec)	No. of mappings	Triples generated	File Size
Article	5.2	17	60589	7.8 MB
Author	5.1	6	121247	14.1 MB
Cell	39.8	10	5940797	645.9 MB
CellFeature	15.4	11	1340610	168.4 MB
CellModule	8.4	2	917367	120.2 MB
CellName	176.2	12	38609232	4.2 GB
CellExecution	8.4	15	125383	48.9 MB
CodeAnalysis	462.4	162	77295103	7.9 GB
Journal	4.7	7	4497	0.49 MB
MarkdownFeature	203.5	125	36693762	3,8 GB
MESH	43.2	3	9078	1.2 MB
Notebook	10.5	24	627127	65.2 MB
NotebookAST	32.0	159	3281939	327.7 MB
NotebookCodeStyle	14.5	4	216105	26.4 MB
NotebookFeature	8.5	29	374071	42.3 MB
NotebookMarkdown	37.5	125	3389551	353.2 MB
NotebookModule	9.4	31	498504	63.3 MB
NotebookName	47.4	115	619562	137.2 MB
Repository	7.1	38	183592	20.1 MB
RepositoryFile	100.6	5	19223736	2.6 GB
RepositoryRelease	6.4	9	252500	33.3 MB
RequirementFile	5.5	6	27865	14.3 MB
Total	1251.7	915	189812217	20.6 GB

With the architecture laid out in Figure 2 and Table 1, it becomes possible to interrogate the dataset about any of the entities, classes or their relationships in a granular fashion. An example query is provided in Figure 2, which asks for notebooks where the reproducibility run produced results identical to those reported in the original publication. It then sorts this set of notebooks by the number of cells and the execution duration of the notebook (both of these parameters can serve as a proxy for the complexity of the notebook itself or the computations triggered by it), and then it limits the results to 10.

In the following, we illustrate the diversity of possible queries by presenting three sets of further example queries. First, Table 2 shows queries corresponding to some figures and tables from the publication describing the original dataset [71]. Second, Table 3 shows a brief selection of other queries that can be queried over the FAIR Jupyter graph. Third, federated queries can be run that combine information from our KG with other KGs, and some examples of such federated queries are given in Table 4. A more comprehensive list of queries is available at

Listing 2 Example query (for live version, see Table 3): Ten successfully reproduced Jupyter notebooks associated with articles indexed in PubMed Central.

```
# Ten successfully reproduced Jupyter notebooks associated with articles indexed in PubMed Central
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX repr: <https://w3id.org/reproduceme/>
PREFIX p-plan: <http://purl.org/net/p-plan>
PREFIX pav: <http://purl.org/pav/>

SELECT DISTINCT
?notebook_url ?total_cells ?duration
WHERE {
  ?execution a repr:CellExecution ;
    p-plan:isStepOfPlan ?notebook ;      # notebook that was executed
    repr:duration ?duration .            # execution duration in seconds
  ?execution repr:processed ?processed_same_result .
    FILTER ((xsd:integer(?processed_same_result) = 51)) # identical results
  ?notebook pav:retrievedFrom ?repository ;              # repo with this notebook
        rdfs:label ?notebook_label ;   # notebook filename
     repr:total_cells ?total_cells .            # number of cells in notebook
  ?repository repr:url ?repo_url_base . # find repo on GitHub
  # create clickable link to notebook on GitHub
  BIND(
  URI(CONCAT( ?repo_url_base, "/blob/main/", ENCODE_FOR_URI(?notebook_label)))
  AS ?notebook_url)
}
# sort by number of cells, then duration, both in descending order
ORDER BY DESC(xsd:float(?total_cells)) DESC(xsd:float(?duration))
LIMIT 10    # limit the output to 10 results
```

Table 2 SPARQL queries to the KG that reproduce materials from the original manuscript describing the dataset [71].

Figure no. in [71]	**SPARQL query**
Fig. 3	Research articles by research field
Fig. 4	Research field (MeSH terms) by the number of GitHub repositories that contain at least one Jupyter notebook.
Fig. 5	Journals with the highest number of articles that had a valid GitHub repository and at least one Jupyter notebook.
Fig. 6	Journals by the number of GitHub repositories and by the number of GitHub repositories with at least one Jupyter notebook.
Fig. 7	Journals by number of GitHub repositories with Jupyter notebooks.
Fig. 9	Programming languages of the notebooks.
Fig. 10	Relative proportion of the most frequent programming languages used in the notebooks per year.
Fig. 11	Python notebooks by minor Python version by year of last commit to the GitHub repository containing the notebook.
Fig. 12	Python notebooks by major Python version by year of first commit to the notebook's GitHub repository.
Fig. 19	Exceptions occurring in Jupyter notebooks in our corpus.
Fig. 22	Jupyter notebook exceptions by research field, taking as a proxy the highest-level MeSH terms of the article associated with the notebook.
Table 2	Notebooks with successful executions with same and different results
Table 4	Common Python code warnings/ style errors in our notebook corpus.

`https://w3id.org/fairjupyter`. Multiple queries with results involving the same entity types can be combined into a profile for that entity type, as described in [51], and we plan on working in this direction. In addition, we combined example queries into a Jupyter notebook, through which users can explore the queries and results, all available at [73], along with a notebook that demonstrates some simple benchmarking measurements for our KG, which are shown in Figure 3.

6 Discussion

In this work, we have demonstrated how the accessibility of an already openly and FAIRly shared dataset can be further improved by leveraging semantic web approaches for representing the data in a KG that can be readily explored from a web browser. That dataset had been created using a workflow for reproducing Jupyter notebooks from biomedical publications, and in the present work, we have enhanced it by representing the dataset's components – publications, GitHub repositories, Jupyter notebooks and reproducibility aspects thereof – using web ontologies.

Besides the central point of of using KGs to increase the FAIRness of a dataset, the present manuscript has a number of other features that are rare or novel in the context of KGs – especially in combination – yet likely of interest to the KG community: (a) using a KG to reproduce a paper's figures and tables, (b) using a Jupyter notebook to document KG queries and results, (c) systematic archiving of in-scope software on Software Heritage, (d) paying attention to a plurality of both programming and natural languages (Julia next to Python, Malayalam in addition to English), (e) environmental footprint assessment, (f) ethics statement. While the latter two also apply to our initial manuscript, a-d do not.

Table 3 General queries over the FAIR Jupyter graph.

Description	SPARQL
Ten successfully reproduced notebooks, as per Figure 2	Query URL
Notebooks by search term: “immun” AND (“stem” OR “differentiation”)	Query URL
Article by keywords, e.g., “open source”	Query URL
Most common errors in immunology	Query URL
Most common errors in Nature journal	Query URL
MeSH terms ranked by “ModuleNotFoundError” frequency	Query URL
GitHub repositories by their stargazers count	Query URL
GitHub repositories and their Software Heritage snapshot	Query URL
Articles and repositories with notebooks in Julia	Query URL

Table 4 Federated queries between FAIR Jupyter and external KGs. Wikidata [78] operates a SPARQL endpoint at `https://query.wikidata.org/`, while MaRDI – the Mathematical Research Data Initiative [23] – operates one at `https://query.portal.mardi4nfdi.de/`.

Description	SPARQL
Match articles between FAIR Jupyter and Wikidata via DOI	Query URL
Match articles between FAIR Jupyter and Wikidata via PMC ID	Query URL
Match articles between FAIR Jupyter and Wikidata via MeSH in a different language (here: Malayalam)	Query URL
Match articles between FAIR Jupyter and MaRDI via DOI and get co-used software	Query URL

The dataset is of particular interest for trainings and education as well as for showcasing real-world examples of actual research practices, from which both best practices and things to avoid can be synthesized. In order to enhance the dataset's usefulness in such contexts, we have paid special attention to its Findability, Accessiblity, Interoperability and Reusability, as per the FAIR Principles [80] for sharing research data. Since our data is about software, we also took into account adaptations of the FAIR Principles to software [44].

The original dataset as a whole was already well aligned with the FAIR Principles, and we added an additional layer of alignment by sharing individual elements of the original dataset in a FAIR way, so as to make it easier for humans and machines to engage with them. At a very basic level, the original dataset as a whole becomes more findable by virtue of the current manuscript and the FAIR Jupyter website pointing to it, more accessible by being available in additional formats (e.g. RDF), more interoperable by compatibility with additional standards (particularly ontologies, as per Figure 2) and more reusable through combinations of the above as well as the enhanced granularity. At a more profound level, the dataset's individual facets and features – be it the classes shown in Figure 2 or the entities listed in Table 1 or any of the relationships between them – have become more FAIR, as they can now be searched, explored, aggregated, filtered and reused in additional user friendly ways both individually or in various combinations, both manually and programmatically. This KG approach facilitates additional routes for engaging with the original data and makes it easy to ask and address questions that were not included in the paper describing the original dataset, hence encouraging readers to build on our work. It also provides an additional level of reproducibility in that it allows for reproducing specific aspects of the original paper, as demonstrated with Table 2. Finally, the coupling of the KG with a

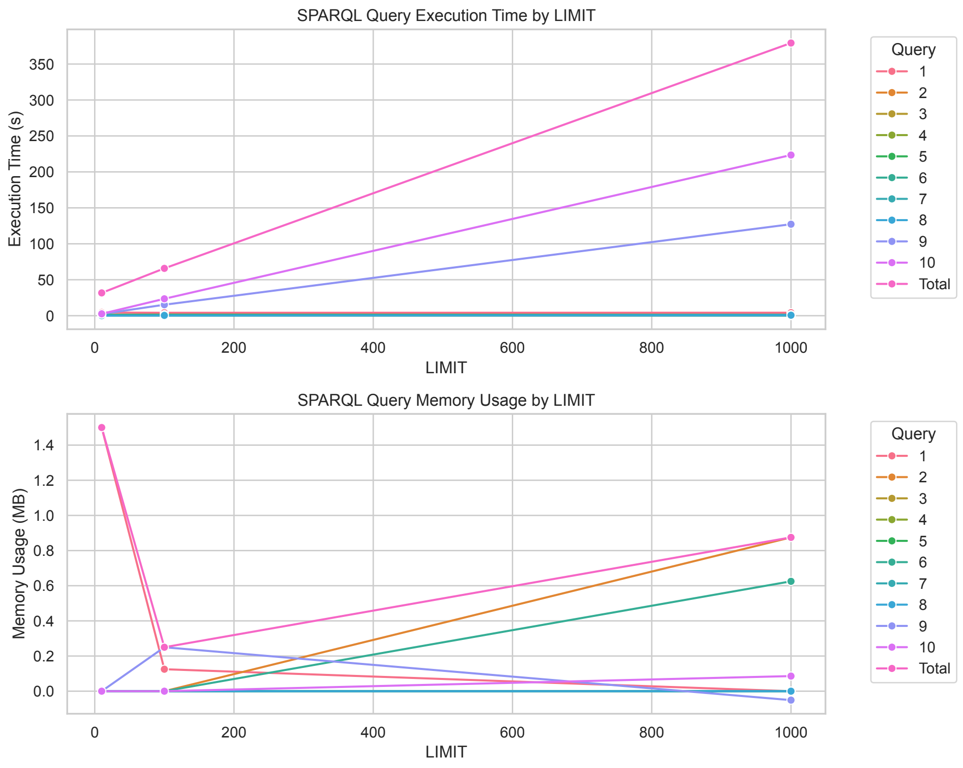

Figure 3 Demonstration of performance measurements of the FAIR Jupyter SPARQL endpoint. Execution time (top) and memory usage (bottom) for a set of 10 SPARQL queries that have been taken from Tables 2, 3 and 4 and were run with different LIMIT values (10, 100, 1000). These measurements were run with a Jupyter notebook that is available at [73], along with the respective query results. With such a simple setup – and perhaps some additional measures like randomized order to reduce position-related artifacts – queries could be identified that would be suitable for assessing performance of the system.

.

public-facing web frontend, a visual graph browser and user-friendly example queries essentially turns the dataset from FAIR data into a FAIR service of potential utility in both research and education settings.

Enhancing a dataset this way and setting up such a KG service represents an additional effort in terms of data sharing. We cannot say at this point whether those efforts – which we have tried to outline in this manuscript – merit the actual benefits in our case, yet our example queries outline some of the potential benefits. We are factoring usage assessment und user feedback into our plans guiding future development of the platform. We are welcoming others to experiment with our workflows or to collaborate with us to adapt them to their use cases.

Assuming some level of usefulness of a service like FAIR Jupyter, keeping it useful requires additional steps like continued maintenance as well as updates in a timely manner. When we ran our Jupyter reproducibility pipeline that resulted in the original dataset, our search query in PubMed Central (cf. 4.1) yielded 3467 articles in March 2023, as opposed to 1419 in February 2021, 5126 in April 2024 and 5941 in October 2024. This translates into an average of several such articles being indexed in PubMed Central per day. We are working on establishing a pipeline that would regularly feed new articles and their notebooks into our KG and on highlighting the successful reproductions, e.g. by badges [42] displayed next to them or through nanopublications [17] sharing reproducibility information about them.

We are working towards extending the KG in other ways, too. For instance, we are exploring to include – or to federate – information about institutions and citations pertaining to articles with Jupyter notebooks, perhaps together with information about resources used alongside Jupyter (e.g. as per Figure 29 in [71] or the MaRDI query in Table 4). We are also conscious that our way of using MeSH terms – which are assigned by PubMed at an article level – does not necessarily represent an optimal way to associate topics with notebooks, for which approaches like a BERTopic [32] analysis of the notebooks themselves might be more suitable.

Furthermore, we could combine FAIR Jupyter with a ReproduceMeGit [66] service that would take Jupyter notebooks as input, assess their reproducibility (potentially even before the corresponding article is submitted to a publication venue) and – depending on the outcome – invoke the KG to suggest similar notebooks (e.g. based on dependencies or topics) with better reproducibility, better documentation or fewer styling issues as a mechanism to help notebook authors familiarize themselves with best practices. Likewise, the FAIR Jupyter KG would provide fertile ground for automated approaches at studying or enhancing computational reproducibility [16, 75].

The reproducibility of any particular notebook is not set in stone, and research questions could be formed around that (e.g. which dependencies contribute most to reproducibility decay, and how that changes over time), which an updated KG could help address, perhaps seeded by incorporating data from our initial run of the pipeline in 2021. This opens up questions around how best to represent time series data in KGs, e.g. as per [20]. Other lines of research could look at deep dependencies [50] of the Jupyter notebooks or their repositories and relate these to aspects of reproducibility – as discussed here – or of security, e.g. as per [47]. As our pipeline is automated, it lends itself to further integration with other automated workflows, for example in the context of workflow systems (e.g. [29]) or machine-actionable data management plans (e.g. [49]).

In its current state, FAIR Jupyter can already support various uses in educational settings, be that the identification of articles to choose for reprohack events [35], materials for lessons by The Carpentries [58], for general reproducibility activities in libraries [31] or for self-guided study by anyone wishing to learn about Jupyter, Python, software dependencies or computational reproducibility. These use cases could be expanded to include, for instance, correlates of reproducibility with individual Python modules or their versions.

We welcome contributions from others – including from other disciplines (e.g. as per [82]) – to the reproducibility pipeline, to the KG or to any of their suggested improvements or applications.

7 Supplementary Material Statement

The FAIR Jupyter service with links to the SPARQL endpoint, code and all the corresponding resources is available at `https://w3id.org/fairjupyter` under the GPL-3.0 license. We are committed to keeping it up for five years, i.e. until April 2029. The code used for generating the original dataset is available at `https://github.com/fusion-jena/computational-reproducibility-pmc`. The CSV files, the YARRRML and RML mappings used for constructing the KG are available at `https://github.com/fusion-jena/fairjupyter` and in Zenodo [69].

Environmental footprint

To estimate the environmental impact of our computation, we leveraged a tool from the Green Algorithms project (`http://www.green-algorithms.org/`). This tool calculates the environmental footprint based on hardware configuration, total runtime, and location. To generate the original dataset, the pipeline consumed 373.78 kWh, resulting in a carbon footprint of approximately

126.58 kg CO2e, equivalent to 11.51 tree years when using default values for Germany. The pipeline for constructing the KG took 20.8 minutes, resulting in a carbon footprint of 7.33 g CO2e. The carbon footprint of the query execution from Table 2 is around 151.48mg CO2e.

Ethical considerations

The original dataset contained the email addresses of the corresponding authors, which are available from PubMed Central as part of the respective article's full text. We have not included these author email addresses within the publicly available KG, since the increased accessibility of the data in the graph also increases the potential for misuse. However, we retain these emails internally to facilitate communication with authors regarding their repositories and the reproducibility of their work.

8 Impact of the Resource

This emerging resource which enhances reproducibility and facilitates information retrieval through KGs, has the potential for impact not just in the Semantic Web community but also other domain-specific communities that extensively work with Jupyter notebooks. Its methodologies are transferable to other domains, making it pertinent to the broader KG community. A bottleneck of our workflow is the availability of full-text access to research articles and the permission to mine them. These conditions are met for biomedicine through the PubMed Central platform that we used, and it provided us with JATS XML, which is straightforward to mine. In principle, our workflow could be adapted to use any other such full-text component instead or in addition. This seems technically doable for Biodiversity PMC [2], an initiative at the Swiss Institute of Bioinformatics that builds on the efforts of PubMed Central and enriches it with additional articles and other information from the biodiversity domain. Another full-text repository amenable to mining is of course arXiv [1], where much of the KG literature can be found already. In both cases, the number of additional Jupyter notebooks that the modified workflow would yield is small relative to the route we chose, but we will keep an eye on the respective developments. FAIR Jupyter can be leveraged for executing federated queries with other resources like Wikidata, as demonstrated in Table 4. This could be expanded further: for instance, one could retrieve additional information such as demographics for papers, authors, affiliations, and details about software packages – see [46] for some gender-based examples. Given the widespread use of Jupyter notebooks across various disciplines in data science, including KG applications, this resource targets a central artifact critical to computational research. It could serve as a guideline for developing recommendations and best practices for creating, maintaining, and executing computational notebooks [56, 41] or setting up services around that, e.g. as per [34]. The resource also offers opportunities to uncover new insights from the dataset that we have not yet explored. These could include, for instance, institutional rankings, or research background of the people involved in successful replications vs. replication problems.

We aim to distill insights from this study into recommendations and infrastructure that enhance the reproducibility of Jupyter notebooks and facilitate the validation of fundamental reproducibility standards. Addressing this challenge requires continued engagement from users and providers of computational resources. We are investigating ways to integrate our materials, workflows, and findings with educational initiatives such as The Carpentries. We are currently incorporating these materials into our own teaching, e.g. we are using and testing it while teaching a "Management of Scientific Data" course at the University of Jena. We are in touch with organizers of ReproHack events, who are interested in using our graph and pipeline to help event attendees find suitable examples to work on. The participants can collaborate to fix notebooks and submit pull requests to repositories, fostering a community effort to improve reproducibility not only of Jupyter notebooks but also of associated repositories, datasets, and results. In the context of the National Research Data Initiative (NFDI) in Germany, two new projects (Jupyter4NFDI and KGI4NFDI) have recently been approved that will provide Jupyter and KG services to dozens of projects involving hundreds of institutions across Germany. We are involved with both and working on including FAIRJupyter into the education and training materials being prepared and disseminated in this context. Both through [71] and through our own editorial activities, we are in touch with a number of journals about their respective reproducibility efforts, where our pipeline and/or the graph are considered to become either part of those efforts or a reference or to be named as a collection of best practice examples or things to avoid. One of us is involved with reproducibility-focused research projects like TIER2 and also serves on the Steering Committee of the German Reproducibility Network. Last but not least, we have been invited to contribute a piece to the Jupyter community blog. Through all these channels, we are receiving feedback and suggestions for developing the resource further and adapting it to specific use cases. As for running our PMC pipeline continuously, we are currently writing this up as a preregistration study aimed at testing the reproducibility of [71] with data from the full year of 2024. We would pipe those 2024 data into a graph in early 2025 and then make the pipeline available to FAIRJupyter users.

9 Conclusion

In this paper, we present the FAIR Jupyter ontology and knowledge graph designed for the semantic sharing and granular exploration of a computational notebook reproducibility dataset. The proposed FAIR Jupyter dataset and knowledge graph are derived from GitHub-hosted Jupyter notebooks linked to biomedical publications. Using state-of-the-art processes and tools, this resource is made available on the web, adhering to best practices. The resulting KG, accessible via a SPARQL endpoint and a web service, exemplifies how to create such resources using advanced methodologies and tools. Although it primarily focuses on the biomedical domain, the methodologies are broadly applicable, making the resource valuable to the wider KG research community and for KG-related education. The central focus on Jupyter notebooks, a common tool in data science, underscores the resource's relevance across various disciplines. Our paper emphasizes technical documentation to enhance usability for both educational purposes and research. We hope that focus on the intersection between KGs, reproducibility and research software engineering will provide fertile ground for engagement from the respective communities and stimulate interactions between them. An example of further research benefiting from our graph would be reproducibility studies focused on deep dependencies of Jupyter notebooks (zooming in on landscape analyses of deep dependencies like [50]). We also plan to establish a pipeline to regularly add new articles and their notebooks into our KG, while highlighting successful reproductions.

References

1 arXiv. URL: https://arxiv.org/.

2 Biodiversity PMC. URL: https://biodiversitypmc.sibils.org/.

3 DOAP: Description of a project. URL: http://usefulinc.com/ns/doap#.

4 flake8nb. URL: https://github.com/s-weigand/flake8-nb.

5 GitHub REST API. URL: https://docs.github.com/en/rest/guides/getting-started-with-the-rest-api.

6 JupyterCon'23. URL: https://cfp.jupytercon.com/2023/talk/FSMWLQ/.

7 Matey. URL: https://rml.io/yarrrml/matey/.

8 MeSH (Medical Subject Headings). URL: https://www.ncbi.nlm.nih.gov/mesh.

9 MeSH SPARQL Endpoint. URL: https://id.nlm.nih.gov/mesh/sparql.

10 SQLite. URL: https://www.sqlite.org.

11 vis.js. URL: https://visjs.org/.

12 Ibrahim Abdelaziz, Julian Dolby, Jamie P. McCusker, and Kavitha Srinivas. A toolkit for gener-

ating code knowledge graphs. In Anna Lisa Gentile and Rafael Gonçalves, editors, *K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021*, pages 137–144. ACM, 2021. doi:10.1145/3460210.3493578.

13 Julián Arenas-Guerrero, David Chaves-Fraga, Jhon Toledo, María S. Pérez, and Oscar Corcho. Morph-KGC: Scalable knowledge graph materialization with mapping partitions. *Semantic Web*, 15(1):1–20, 2024. doi:10.3233/SW-223135.

14 Sören Auer, Viktor Kovtun, Manuel Prinz, Anna Kasprzik, Markus Stocker, and Maria Esther Vidal. Towards a knowledge graph for science. In *Proceedings of the 8th international conference on web intelligence, mining and semantics*, pages 1–6, 2018. doi:10.1145/3227609.3227689.

15 Kathrin Blagec, Adriano Barbosa-Silva, Simon Ott, and Matthias Samwald. A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks. *Scientific Data*, 9(1):322, 2022.

16 Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. Super: Evaluating agents on setting up and executing tasks from research repositories, 2024. doi:10.48550/arXiv.2409.07440.

17 Cristina-Iulia Bucur, Tobias Kuhn, Davide Ceolin, and Jacco van Ossenbruggen. Nanopublication-based semantic publishing and reviewing: a field study with formalization papers. *PeerJ Computer Science*, 9:e1159, 2023. doi:10.7717/peerj-cs.1159.

18 Jeremy J Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83, 2004. doi:10.1145/1013367.1013381.

19 Souti Chattopadhyay, Ishita Prasad, Austin Z Henley, Anita Sarma, and Titus Barik. What's wrong with computational notebooks? pain points, needs, and design opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020. doi:10.1145/3313831.3376729.

20 Zheyuan Chen, Yuwei Wan, Ying Liu, and Agustin Valera-Medina. A knowledge graph-supported information fusion approach for multi-faceted conceptual modelling. *Inf. Fusion*, 101:101985, 2024. doi:10.1016/j.inffus.2023.101985.

21 Paolo Ciccarese, Stian Soiland-Reyes, Khalid Belhajjame, Alasdair J. G. Gray, Carole A. Goble, and Tim Clark. PAV ontology: provenance, authoring and versioning. *J. Biomed. Semant.*, 4:1–22, 2013. doi:10.1186/2041-1480-4-37.

22 Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009. doi:10.1093/bioinformatics/btp163.

23 The MaRDI consortium. MaRDI: Mathematical Research Data Initiative Proposal, May 2022. doi:10.5281/zenodo.6552436.

24 Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. Rml: A generic language for integrated rdf mappings of heterogeneous data. *Ldow*, 1184, 2014. URL: https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

25 Diego Esteves, Diego Moussallem, Ciro Baron Neto, Tommaso Soru, Ricardo Usbeck, Markus Ackermann, and Jens Lehmann. Mex vocabulary: a lightweight interchange format for machine learning experiments. In *Proceedings of the 11th International Conference on Semantic Systems*, pages 169–176, 2015. doi:10.1145/2814864.2814883.

26 Michael Färber. The microsoft academic knowledge graph: A linked data source with 8 billion triples of scholarly data. In *The Semantic Web – ISWC 2019*, pages 113–129, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-30796-7_8.

27 Michael Färber. Analyzing the github repositories of research papers. In Ruhua Huang, Dan Wu, Gary Marchionini, Daqing He, Sally Jo Cunningham, and Preben Hansen, editors, *JCDL '20: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, Virtual Event, China, August 1-5, 2020*, JCDL '20, pages 491–492, New York, NY, USA, 2020. ACM. doi:10.1145/3383583.3398578.

28 Daniel Garijo and Yolanda Gil. Augmenting PROV with plans in P-PLAN: scientific processes as linked data. In Tomi Kauppinen, Line C. Pouchard, and Carsten Keßler, editors, *Proceedings of the Second International Workshop on Linked Science 2012 - Tackling Big Data, Boston, MA, USA, November 12, 2012*, volume 951 of *CEUR Workshop Proceedings*. CEUR Workshop Proceedings, CEUR-WS.org, 2012. URL: https://ceur-ws.org/Vol-951/paper6.pdf.

29 Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R. Crusoe, Kristian Peters, and Daniel Schober. FAIR Computational Workflows. *Data Intelligence*, 2(1-2):108–121, 2020. doi:10.1162/dint_a_00033.

30 Brian E. Granger and Fernando Perez. Jupyter: Thinking and Storytelling With Code and Data. *Computing in Science and Engineering*, 23(2):7–14, 2021. doi:10.1109/MCSE.2021.3059263.

31 Sabrina Granger. How research reproducibility challenges librarians' skill sets. A French librarian's perspective. *Journal for Reproducibility in Neuroscience*, 2, 2020. https://jrn.trialanderror.org/pub/french-librarians-perspective.

32 Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based TF-IDF procedure. *CoRR*, abs/2203.05794, 2022. doi:10.48550/arXiv.2203.05794.

33 Björn Grüning, John Chilton, Johannes Köster, Ryan Dale, Nicola Soranzo, Marius van den Beek, Jeremy Goecks, Rolf Backofen, Anton Nekrutenko, and James Taylor. Practical Computational Re-

producibility in the Life Sciences. *Cell systems*, 6(6):631–635, 2018.

34 Björn Hagemeier, Arnim Bleier, Bernd Flemisch, Matthias Lieber, Klaus Reuter, and George Dogaru. Jupyter4nfdi, July 2024. doi:10.5281/zenodo.12699382.

35 Kristina Hettne, Ricarda Proppert, Linda Nab, L. Paloma Rojas-Saunero, and Daniela Gawehns. Reprohacknl 2019: how libraries can promote research reproducibility through community engagement. *IASSIST quarterly*, 44(1-2):1–10, 2020.

36 Pieter Heyvaert, Ben De Meester, Anastasia Dimou, and Ruben Verborgh. Declarative rules for linked data generation at your fingertips! In *The Semantic Web: ESWC 2018 Satellite Events: ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers 15*, pages 213–217. Springer, 2018. doi:10.1007/978-3-319-98192-5_40.

37 Nicolas Hubert, Pierre Monnin, Mathieu d'Aquin, Armelle Brun, and Davy Monticolo. Pygraft: Configurable generation of schemas and knowledge graphs at your fingertips. *CoRR*, abs/2309.03685, 2023. doi:10.48550/arXiv.2309.03685.

38 Hassan Hussein, Kheir Eddine Farfar, Allard Oelen, Oliver Karras, and Sören Auer. Increasing reproducibility in science by interlinking semantic artifact descriptions in a knowledge graph. In *International Conference on Asian Digital Libraries*, pages 220–229. Springer, 2023. doi:10.1007/978-981-99-8088-8_19.

39 Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D'Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246, 2019. doi:10.1145/3360901.3364435.

40 Aidan Kelley and Daniel Garijo. A framework for creating knowledge graphs of scientific software metadata. *Quantitative Science Studies*, 2(4):1423–1446, 2021. doi:10.1162/qss_a_00167.

41 Dominik Kerzel, Birgitta König-Ries, and Sheeba Samuel. MLProvLab: Provenance management for data science notebooks. In *Datenbanksysteme für Business, Technologie und Web (BTW 2023), 20. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 06.-10, März 2023, Dresden, Germany, Proceedings*, volume P-331 of *LNI*, pages 965–980. Gesellschaft für Informatik e.V., 2023. doi:10.18420/BTW2023-66.

42 Mallory C Kidwell, Ljiljana Lazarevic, Erica Baranski, Tom E. Hardwicke, Sarah Piechowski, Lina-Sophia Falkenberg, Curtis Kennett, Agnieszka Slowik, Carina Sonnleitner, Chelsey Hess-Holden, Timothy M Errington, Susann Fiedler, and Brian A Nosek. Badges to Acknowledge Open Practices: A Simple, Low-Cost, Effective Method for Increasing Transparency. *PLOS Biology*, 14(5):e1002456, 2016. arXiv:27171007.

43 Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E. Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B. Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter notebooks - a publishing format for reproducible computational workflows. In Fernando Loizides and Birgit Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas, 20th International Conference on Electronic Publishing, Göttingen, Germany, June 7-9, 2016*, pages 87–90. IOS Press, 2016. doi:10.3233/978-1-61499-649-1-87.

44 Anna-Lena Lamprecht, Leyla J. García, Mateusz Kuzak, Carlos Martinez-Ortiz, Ricardo Arcila, Eva Martin Del Pico, Victoria Dominguez Del Angel, Stephanie van de Sandt, Jon C. Ison, Paula Andrea Martínez, Peter McQuilton, Alfonso Valencia, Jennifer L. Harrow, Fotis E. Psomopoulos, Josep Lluis Gelpí, Neil P. Chue Hong, Carole A. Goble, and Salvador Capella-Gutiérrez. Towards FAIR principles for research software. *Data Sci.*, 3(1):37–59, 2020. doi:10.3233/ds-190026.

45 Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. *W3C recommendation*, 30, 2013.

46 Ekaterina Levitskaya, Gizem Korkmaz, Daniel Mietchen, and Lane Rasberry. Analysis of linked github and wikidata, December 2022. doi:10.5281/zenodo.7443339.

47 Mario Lins, René Mayrhofer, Michael Roland, Daniel Hofer, and Martin Schwaighofer. On the critical path to implant backdoors and the effectiveness of potential mitigation techniques: Early learnings from xz, 2024. doi:10.48550/arXiv.2404.08987.

48 Chang Liu, Matthew Kim, Michael Rueschman, and Satya S. Sahoo. *ProvCaRe: A Large-Scale Semantic Provenance Resource for Scientific Reproducibility*, pages 59–73. Springer International Publishing, Cham, 2021. doi:10.1007/978-3-030-67681-0_5.

49 Tomasz Miksa, Stephanie Renee Simms, Daniel Mietchen, and Sarah Jones. Ten principles for machine-actionable data management plans. *PLoS Comput. Biol.*, 15(3):e1006750, 2019. doi:10.1371/journal.pcbi.1006750.

50 Andrew Nesbitt, Boris Veytsman, Daniel Mietchen, Eva Maxfield Brown, James Howison, João Felipe Pimentel, Laurent Hébert-Dufresne, and Stephan Druskat. Biomedical open source software: Crucial packages and hidden heroes. *CoRR*, abs/2404.06672, 2024. doi:10.48550/arXiv.2404.06672.

51 Finn Årup Nielsen, Daniel Mietchen, and Egon L. Willighagen. Scholia, scientometrics and wikidata. In Eva Blomqvist, Katja Hose, Heiko Paulheim, Agnieszka Lawrynowicz, Fabio Ciravegna, and Olaf Hartig, editors, *The Semantic Web: ESWC 2017 Satellite Events - ESWC 2017 Satellite Events, Portorož, Slovenia, May 28 - June 1, 2017, Revised Selected Papers*, volume 10577 of *Lecture Notes in Computer Science*, pages 237–259. Springer, 2017. doi:10.1007/978-3-319-70407-4_36.

52 Daniel Nüst, Vanessa V. Sochat, Ben Marwick, Stephen J. Eglen, Tim Head, Tony Hirst, and Benjamin D Evans. Ten simple rules for writing

Dockerfiles for reproducible data science. *PLOS Computational Biology*, 16(11):e1008316, 2020. doi:10.1371/journal.pcbi.1008316.

53 Jeff Z Pan. Resource description framework. In *Handbook on ontologies*, pages 71–90. Springer, 2009. doi:10.1007/978-3-540-92673-3_3.

54 Silvio Peroni and David Shotton. Fabio and cito: Ontologies for describing bibliographic resources and citations. *Journal of Web Semantics*, 17:33–43, 2012. doi:10.1016/j.websem.2012.08.001.

55 João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. A large-scale study about quality and reproducibility of jupyter notebooks. In *Proceedings of the 16th International Conference on Mining Software Repositories*, MSR '19, pages 507–517, Piscataway, NJ, USA, 2019. IEEE Press. doi:10.1109/MSR.2019.00077.

56 João Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. Understanding and improving the quality and reproducibility of jupyter notebooks. *Empir. Softw. Eng.*, 26(4):65, 2021. doi:10.1007/s10664-021-09961-9.

57 Project Jupyter. nbdime: Jupyter notebook diff and merge tools. https://github.com/jupyter/nbdime, 2021. Accessed 22 November 2024.

58 Sarah Pugachev. What are "the carpentries" and what are they doing in the library? *portal: Libraries and the Academy*, 19(2):209–214, 2019.

59 Richard J. Roberts. Pubmed central: The genbank of the published literature. *Proceedings of the National Academy of Sciences*, 98(2):381–382, 2001.

60 A Rule, A Birmingham, C Zuniga, I Altintas, SC Huang, R Knight, N Moshiri, MH Nguyen, SB Rosenthal, F Pérez, et al. Ten simple rules for writing and sharing computational analyses in jupyter notebooks. *Plos Computational Biology*, 15(7):e1007007–e1007007, 2019. doi:10.1371/journal.pcbi.1007007.

61 Adam Rule, Aurélien Tabard, and James D. Hollan. Exploration and explanation in computational notebooks. In Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox, editors, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, CHI '18, page 32, New York, NY, USA, 2018. ACM. doi:10.1145/3173574.3173606.

62 Satya S Sahoo, Joshua Valdez, Michael Rueschman, and Matthew Kim. Semantic provenance graph for reproducibility of biomedical research studies: Generating and analyzing graph structures from published literature. In *MEDINFO 2019: Health and Wellbeing e-Networks for All*, pages 328–332. IOS Press, 2019. doi:10.3233/SHTI190237.

63 Angelo A Salatino, Thiviyan Thanapalasingam, Andrea Mannocci, Francesco Osborne, and Enrico Motta. The computer science ontology: a large-scale taxonomy of research areas. In *The Semantic Web–ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8–12, 2018, Proceedings, Part II 17*, pages 187–205. Springer, 2018. doi:10.1007/978-3-030-00668-6_12.

64 Sheeba Samuel and Birgitta König-Ries. Combining P-Plan and the REPRODUCE-ME ontology to achieve semantic enrichment of scientific experiments using interactive notebooks. In *The Semantic Web: ESWC 2018 Satellite Events: Heraklion, Crete, Greece, June 3-7, 2018*, pages 126–130. Springer, 2018. doi:10.1007/978-3-319-98192-5_24.

65 Sheeba Samuel and Birgitta König-Ries. ProvBook: Provenance-based semantic enrichment of interactive notebooks for reproducibility. In *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, volume 2180 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2018. URL: https://ceur-ws.org/Vol-2180/paper-57.pdf.

66 Sheeba Samuel and Birgitta König-Ries. ReproduceMeGit: A visualization tool for analyzing reproducibility of jupyter notebooks. In *Provenance and Annotation of Data and Processes*, pages 201–206, Cham, 2021. Springer International Publishing. doi:10.1007/978-3-030-80960-7_12.

67 Sheeba Samuel and Birgitta König-Ries. End-to-end provenance representation for the understandability and reproducibility of scientific experiments using a semantic approach. *Journal of biomedical semantics*, 13(1):1, 2022. doi:10.1186/s13326-021-00253-1.

68 Sheeba Samuel and Daniel Mietchen. FAIR Jupyter. Service, DFG 514664767, DFG 460135501, DFG 521453681 (visited on 2024-11-29). URL: https://w3id.org/fairjupyter, doi:10.4230/artifacts.22527.

69 Sheeba Samuel and Daniel Mietchen. FAIR Jupyter Knowledge Graph: v1.0. Software, version 1.0., DFG 514664767, DFG 460135501, DFG 521453681 (visited on 2024-11-29). doi:10.5281/zenodo.14197755.

70 Sheeba Samuel and Daniel Mietchen. Dataset of a Study of Computational reproducibility of Jupyter notebooks from biomedical publications. https://doi.org/10.5281/zenodo.8226725, 2023.

71 Sheeba Samuel and Daniel Mietchen. Computational reproducibility of Jupyter notebooks from biomedical publications. *GigaScience*, 13:giad113, 2024.

72 Sheeba Samuel and Daniel Mietchen. FAIR Jupyter Knowledge Graph, September 2024. doi:10.5281/zenodo.13845701.

73 Sheeba Samuel and Daniel Mietchen. FAIR Jupyter Knowledge Graph: SPARQL Queries and Performance Evaluation and Benchmark, September 2024. doi:10.5281/zenodo.13847627.

74 Max Schröder, Frank Krüger, and Sascha Spors. Reproducible research is more than publishing research artefacts: A systematic analysis of jupyter notebooks from research articles. *CoRR*, abs/1905.00092, 2019. arXiv:1905.00092, doi:10.48550/arXiv.1905.00092.

75 Zachary S. Siegel, Sayash Kapoor, Nitya Nagdir, Benedikt Stroebl, and Arvind Narayanan. Core-bench: Fostering the credibility of published

research through a computational reproducibility agent benchmark, 2024. doi:10.48550/arXiv.2409.11363.

76 Mari Carmen Suárez-Figueroa, Asunción Gómez-Pérez, and Mariano Fernández-López. The neon methodology for ontology engineering. In *Ontology engineering in a networked world*, pages 9–34. Springer, 2011. doi:10.1007/978-3-642-24794-1_2.

77 Ana Trisovic, Matthew K Lau, Thomas Pasquier, and Mercè Crosas. A large-scale study on research code quality and execution. *Scientific Data*, 9(1):60, 2022.

78 Denny Vrandečić, Lydia Pintscher, and Markus Krötzsch. Wikidata: The making of. In Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben, editors, *Companion Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pages 615–624. ACM, 2023. doi:10.1145/3543873.3585579.

79 Jiawei Wang, Tzu-yang Kuo, Li Li, and Andreas Zeller. Restoring reproducibility of jupyter notebooks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 288–289, 2020. doi:10.1145/3377812.3390803.

80 Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

81 Alistair Willis, Patricia Charlton, and Tony Hirst. Developing students' written communication skills with jupyter notebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1089–1095, 2020. doi:10.1145/3328778.3366927.

82 Morgan F. Wofford, Bernadette M. Boscoe, Christine L. Borgman, Irene V. Pasquetto, and Milena S. Golshan. Jupyter notebooks as discovery mechanisms for open science: Citation practices in the astronomy community. *Computing in Science & Engineering*, 22(1):5–15, 2020. doi:10.1109/MCSE.2019.2932067.

83 Jian Xu, Sunkyu Kim, Min Song, Minbyul Jeong, Donghyeon Kim, Jaewoo Kang, Justin F Rousseau, Xin Li, Weijia Xu, Vetle I Torvik, et al. Building a pubmed knowledge graph. *Scientific data*, 7(1):205, 2020.

Table 5 The FAIR Jupyter Ontology Requirements Specification Document.

The FAIR Jupyter Ontology Requirements Specification Document	
1. Purpose	
The purpose of this ontology is to provide a knowledge model for analyzing, categorizing, and understanding the reproducibility, dependencies, errors, and metadata of Jupyter notebooks cited in academic research.	
2. Scope	
The ontology has to focus on the computational and non-computational processes of an experiment and the data used and generated in an experiment. The level of granularity is directly related to the competency questions and terms identified	
3. Implementation Language	
The ontology will be implemented in the Web Ontology Language (OWL).	
4. Intended End-Users	
User 1. Researchers and academics conducting reproducibility studies. User 2. Data scientists and developers analyzing Jupyter notebooks. User 3. Journals and publishers interested in ensuring the reproducibility of published research. User 4. Educational institutions integrating Jupyter notebooks into curricula. User 5. Software tools and platforms providing support for notebook analysis and management.	
5. Intended Uses	
Use 1. Facilitating reproducibility studies by providing a standardized framework for analyzing notebook execution and outcomes. Use 2. Identifying common errors, dependencies, and issues in Jupyter notebooks to improve their reliability. Use 3. Enabling the integration of best practices and coding standards compliance checks in Jupyter notebooks. Use 4. Providing a basis for educational and training materials on best practices for using Jupyter notebooks in research.	
6. Ontology Requirements	
a. Non-Functional Requirements	
NFR 1. The ontology must be published on the Web with an open license. NFR 2. The ontology must reuse other ontologies if required.	
b. Functional Requirements: Groups of Competency Questions	
CQG1. Journals	CQG2. Research Fields
CQ1. What is the timeline of journals (by year) that publish their first and tenth article mentioning a GitHub repository with a Jupyter/iPython notebook? CQ2. Which journals have significantly higher or lower reproducibility rates?	CQ3. What are the top-level MeSH terms as a proxy for their research field ranked by the number of GitHub repositories, and which of these repositories contain at least one Jupyter notebook? CQ4. How does the error rate in notebooks differ by research field?
CQG3. Articles	CQG4. Repositories
CQ5. Are notebooks associated with more recent articles more or less likely to have certain errors? CQ6. How does the error rate in notebooks differ between the top and bottom 10th percentiles of articles? CQ7. How many repositories are mentioned in the articles? CQ8. What is the timeline of articles (by year) that mention a GitHub repository with a Jupyter/iPython notebook? CQ9. What is the timeline of subjects (by year) that publish their first and tenth article mentioning a GitHub repository with a Jupyter/iPython notebook?	CQ10. What percentage of repositories mentioned in articles are still available at the indicated GitHub address? CQ11. What percentage of repositories have official releases? CQ12. What is the number of collaborators, forks, and stars for each repository? CQ13. How many commits have been made to the repository after the publication of the article? CQ14. Which repositories contain declared dependencies (e.g., requirements.txt, setup.py, or Pipfile)? CQ15. What is the percentage of Jupyter code compared to the overall code in the repository? CQ16. How many repositories do not contain any notebooks?
CQG5. Notebooks	CQG6. Programming Languages
CQ17. How many notebooks are PEP8 compliant? CQ18. What are the most popular modules/libraries used in the notebooks? CQ19. What are the most popular APIs used in the notebooks? CQ20. How many notebooks are in repositories that contain a requirements.txt, setup.py, or Pipfile? CQ21. How many notebooks are shared in the actual order of execution of their cells? CQ22. How many cells are code cells? CQ23. How many cells are Markdown cells? CQ24. Which notebooks have the highest ratio of Markdown cells to code cells?	CQ25. Are there groupwise differences between R and Python notebooks/repositories in terms of errors and reproducibility? CQ26. What "nbformat" versions are used in the notebooks? CQ27. How many notebooks use "nbformat" version 3 or lower? CQ28. What are the top 15 programming languages used in the notebooks? CQ29. How many Python notebooks are valid or invalid? CQ30. How many notebooks have an undeclared programming language? CQ31. What versions of Python are used in the notebooks?

Table 6 The FAIR Jupyter Ontology Requirements Specification Document.

CQG7. Notebook Reproducibility	CQG8. General
CQ32. What are the common data dependencies in the notebooks? CQ33. What types of file errors occur in the notebooks? CQ34. How often do notebooks reference local files? CQ35. What are the best practices for file naming in the notebooks? CQ36. How often are URLs used in the notebooks, and what issues arise from their usage? CQ37. How frequently are persistent identifiers used in the notebooks? CQ38. What types of HTTP errors are encountered in the notebooks? CQ39. What are the common causes of failed dependencies in the notebooks? CQ40. What are the common reasons for installation failures in the notebooks? CQ41. How often do notebooks fail due to missing files? CQ42. How many notebooks finish execution successfully? CQ43. How many notebooks produce different results on re-execution? CQ44. How many notebooks produce the same results on re-execution?	CQ45. Is the usage of other best practices (e.g., some level of CI in the repo, PEP8 compliance in the notebooks, usage of DOIs for article, data, and code, ORCID for authors, or having a data/code availability statement) predictive of reproducibility success? CQ46. How does the presence of best practices correlate with reproducibility success? CQ47. What percentage of authors on a article have an ORCID? CQ48. What is the environmental footprint of reproducing a Jupyter notebook? CQ49. What is the distribution by country of author affiliations for the notebooks?

7. Pre-Glossary of Terms

a. Terms from Competency Questions + Frequency

Notebook 37 Article 10 Code 7 Fail 3 Journal 2	Repository 15 GitHub 5 Python 6 File 6 Module 1	Dependencies 3 Cell 6 Execution 5 Reproducibility 4 Author 3	Error 6 Programming language 2 Research field 2 Markdown 2 Release 1

b. Terms from Answers + Frequency

Notebook 37 Article 10 Code 7 Fail 3 Journal 2	Repository 15 GitHub 5 Python 6 File 6 Module 1	Dependencies 3 Cell 7 Execution 5 Reproducibility 4 Author 3	Error 6 Programming language 2 Research field 2 Markdown 2 Release 1

c. Objects

No objects were identified.