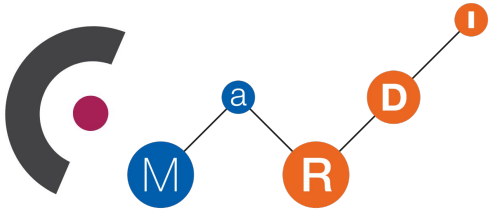
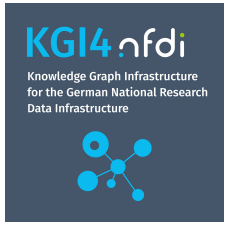
Fine-grained exploration of the reproducibility of research-related Jupyter notebooks at scale

Daniel Mietchen (FIZ Karlsruhe & MaRDI & KGI4NFDI) & Sheeba Samuel (TU Chemnitz)



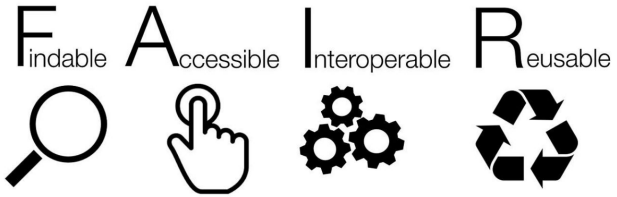


deRSE25, 26 February 2025 - DOI: 10.5281/zenodo.14678185

Outline

- Motivation
- FAIR Jupyter
- Results
- Future Work

Motivation

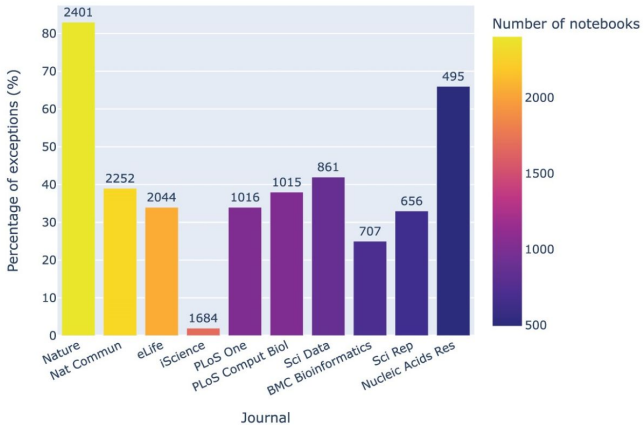


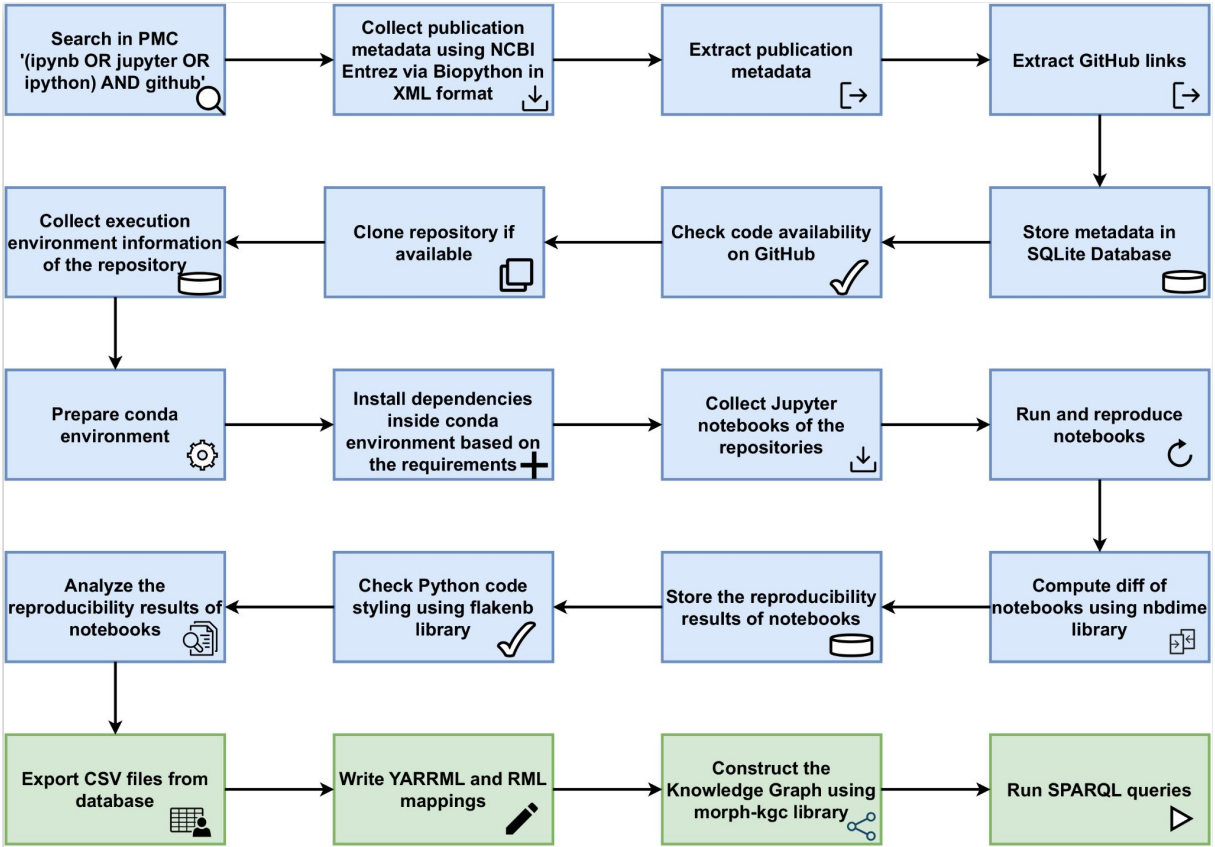
- Enhancing Research & Education:
  - Provide example queries for diverse research software engineering and educational use cases.
- Profiling Content Types:
  - Demonstrate how such query sets can profile content by type or class.
- FAIR Principles:
  - Improve findability, accessibility, interoperability, and reusability of datasets.
- Semantic Data Sharing:
  - Enable better identification, communication, and reuse of complex datasets.
- Quality & Best Practices:
  - Support standardization, reproducibility, and best practices in research software.

FAIR Jupyter Overview

190 million triples

740	3,467	2,660	27,271
Journals	Publications	Repositories	Notebooks

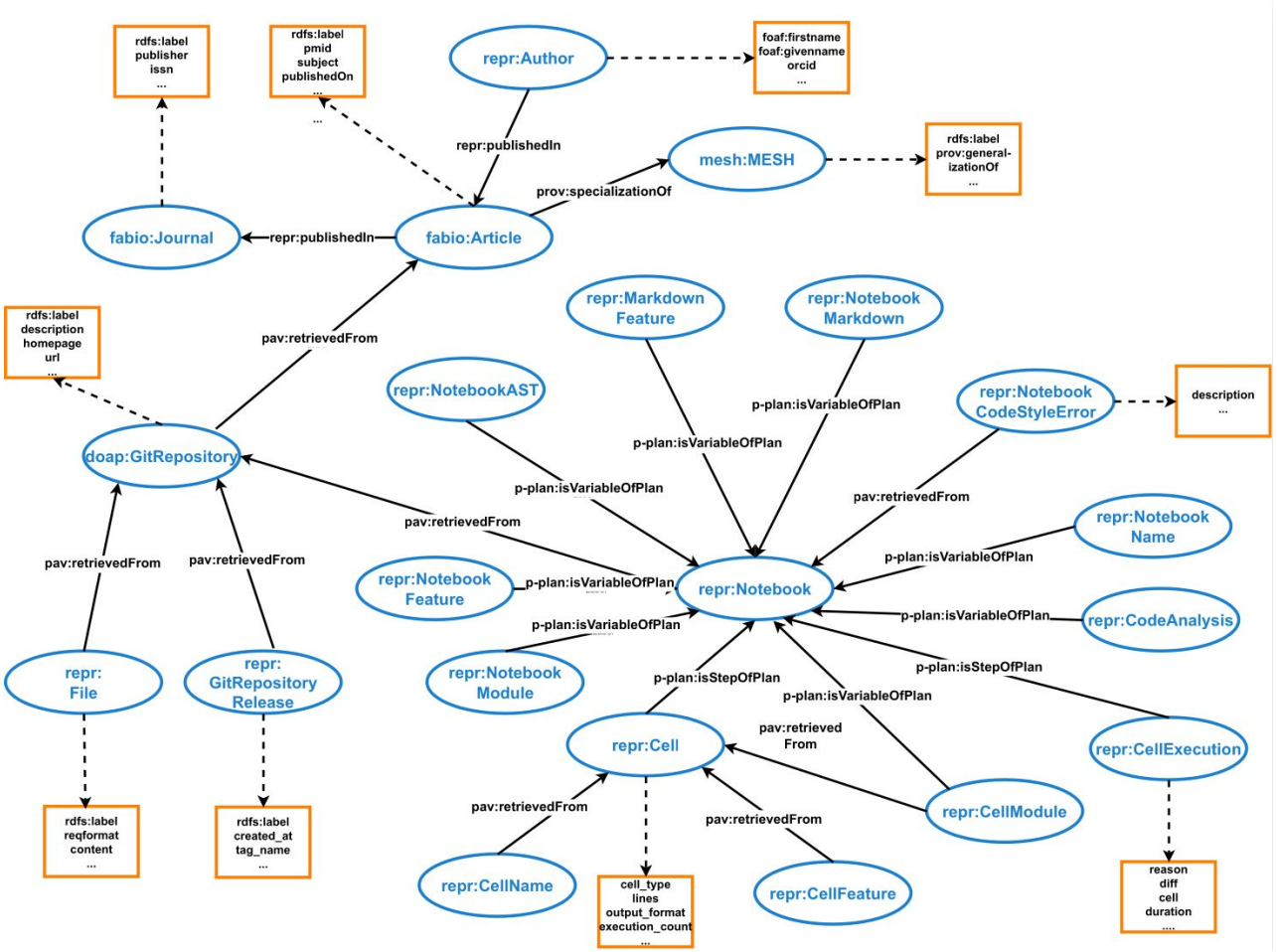




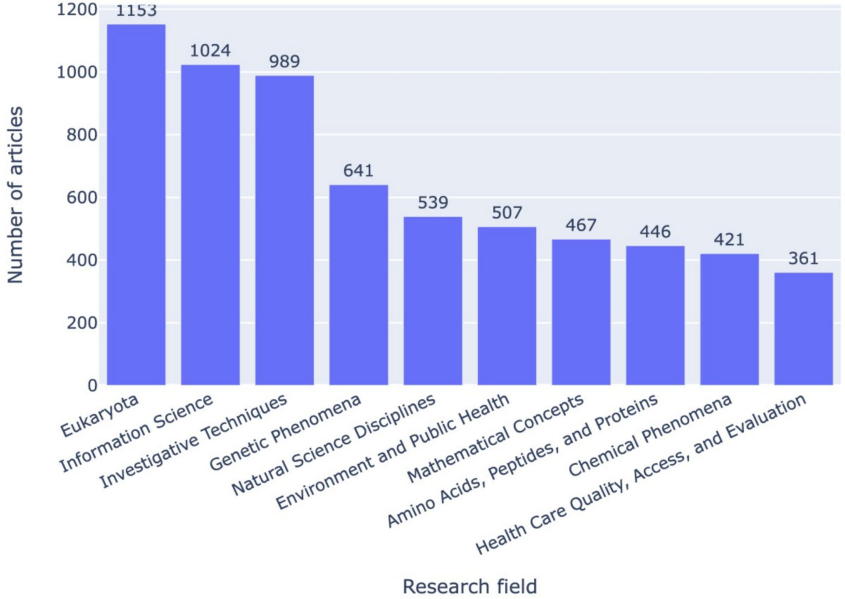
FAIR Jupyter

- A knowledge graph approach to semantic sharing and granular exploration of a computational notebook reproducibility dataset
- Initial Run
  - 2021
  - Preprint: https://doi.org/10.48550/arXiv.2209.04308
- Rerun
  - 2023
  - Code : https://github.com/fusion-jena/computational-reproducibility-pmc
  - Data : https://doi.org/10.5281/zenodo.8226725
  - Paper : https://doi.org/10.1093/gigascience/giad113 (see also SE25 session today at 11:00)
- Knowledge graph
  - Website: https://w3id.org/fairjupyter
  - SPARQL Endpoint: https://reproduceme.uni-jena.de/#/dataset/fairjupyter/query
  - Knowledge Graph Browser: https://reproduceme.uni-jena.de/fj/
  - Sample query: PLOS Comp Biology articles and associated GitHub repos with Jupyter notebooks
  - Paper: https://doi.org/10.4230/TGDK.2.2.4

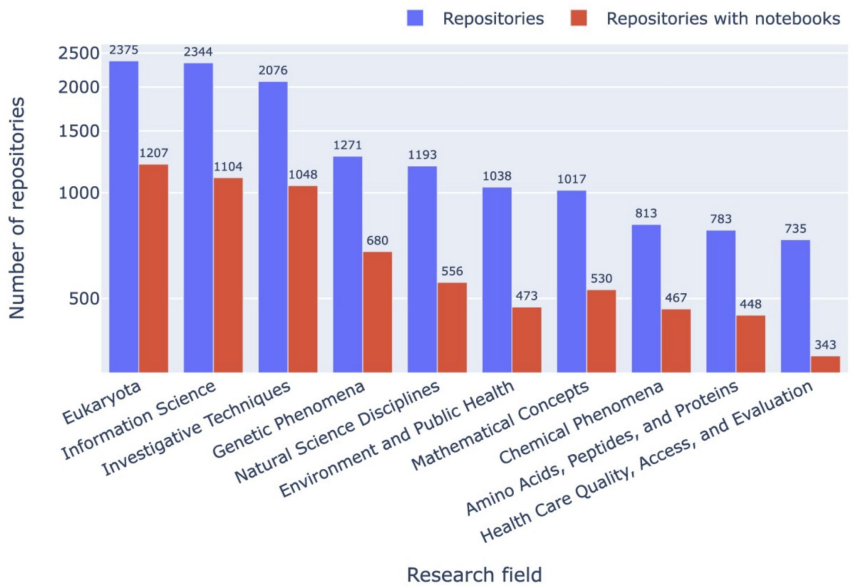
FAIR Jupyter Ontology



Results - Research Field



SPARQL Query: [Research articles by research field]



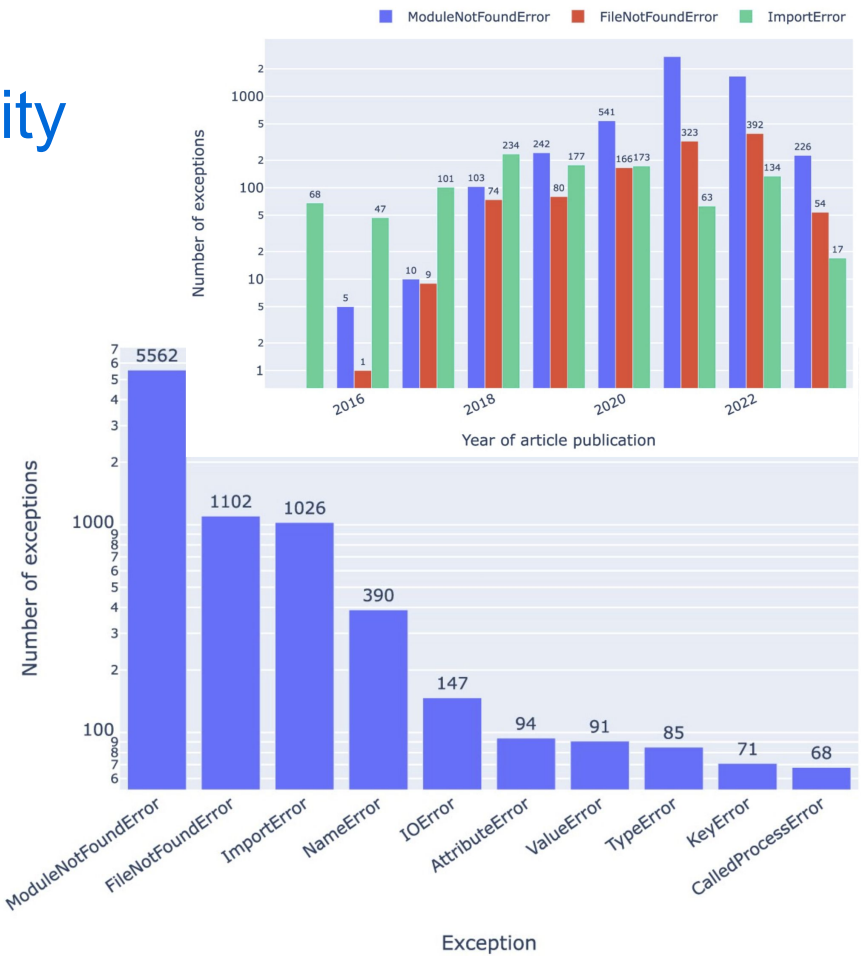
SPARQL Query: [Research field (MeSH terms) by the number of GitHub repositories that contain at least one Jupyter notebook.]

Results - Programming Languages

Notebook language	Rerun	Initial run	Ratio rerun/initial
Matlab	134	9	14.9
Julia	295	59	5.0
Unknown	3,112	720	4.3
Python	22,578	8,160	2.8
R	891	461	1.9
Bash	36	24	1.5
Sos	29	24	1.2
Groovy	95	95	1.0
Scala	29	29	1.0
Java	8	8	1.0

Results - Notebook Reproducibility

Features	Notebooks with `different` results	Notebooks with `identical` results
Number of notebooks	324 (⌀ 151)	879 (⌀ 245)
setup.py	0 (⌀ 0)	344 (⌀ 98)
requirement.txt	1 (⌀ 0)	353 (⌀ 107)
pipfile	0 (⌀ 0)	0 (⌀ 0)
Total cells	23 (⌀ 17.9)	19.6 (⌀ 17.1)
Code cells	15.4 (⌀ 12.3)	11.3 (⌀ 9.8)
Markdown cells	7.6 (⌀ 5.6)	8.3 (⌀ 6.7)
Ratio of Markdown vs. code cells	0.49 (⌀ 0.46)	0.73 (⌀ 0.68)
Empty cells	0.9 (⌀ 0.7)	0.7 (⌀ 0.7)
Differences	6.3 (⌀ 5.3)	0 (⌀ 0)
Execution time (s)	18.3 (⌀ 22.1)	57.6 (⌀ 16.4)
Execution time per code cell (s)	1.88 (⌀ 1.80)	5.09 (⌀ 1.67)



SPARQL Query: Exceptions occurring in Jupyter notebooks in our corpus.

Results - Notebook Reproducibility

SPARQL Endpoint: /fairjupyter/sparql

Content Type (SELECT): JSON

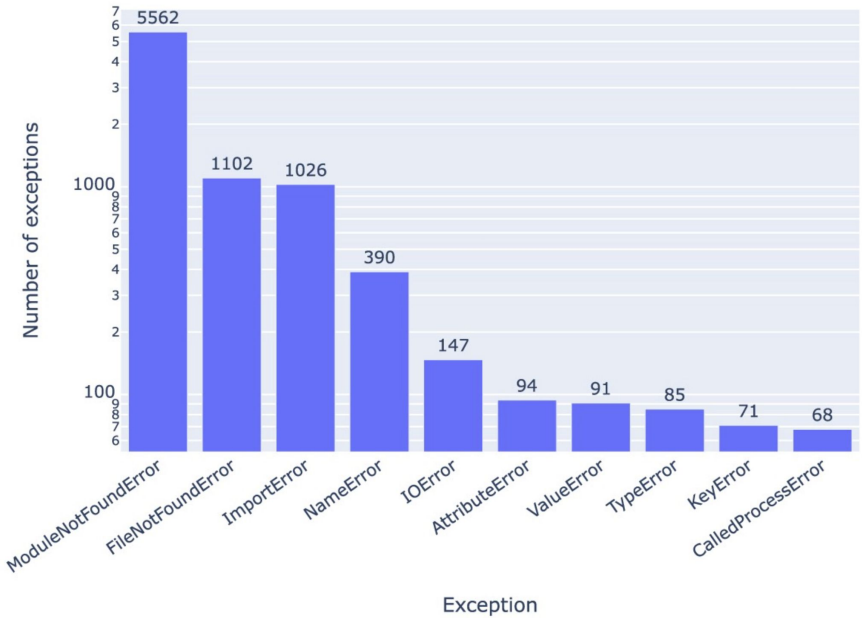
Content Type (GRAPH): Turtle

```
SELECT ?exception (COUNT(?exception) AS ?count)
WHERE {
  ?execution  a <https://w3id.org/reproduceme/CellExecution> ;
    <https://w3id.org/reproduceme/exception> ?exception .
}
GROUP BY ?exception
ORDER BY DESC(?count)
LIMIT 10
```

Table | Response | 10 results in 0.169 seconds | Simple view | Ellipse | Filter query results | Page size: 50

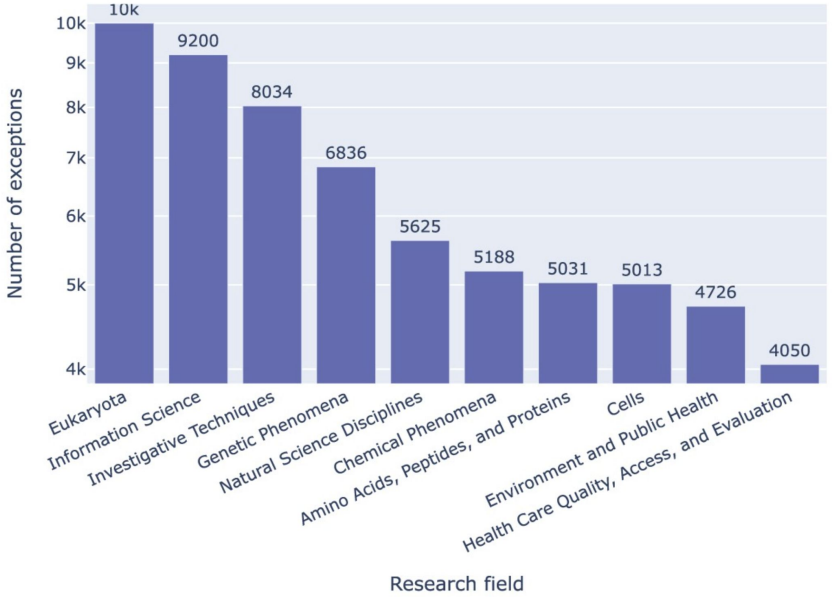
	exception	count
1	ModuleNotFoundError	"5562"^^<http://www.w3.org/2001/XMLSchema
2	FileNotFoundError	"1102"^^<http://www.w3.org/2001/XMLSchema
3	ImportError	"1026"^^<http://www.w3.org/2001/XMLSchema
4	NameError	"390"^^<http://www.w3.org/2001/XMLSchema#
5	IOError	"147"^^<http://www.w3.org/2001/XMLSchema#
6	AttributeError	"94"^^<http://www.w3.org/2001/XMLSchema#in
7	ValueError	"91"^^<http://www.w3.org/2001/XMLSchema#in
8	TypeError	"85"^^<http://www.w3.org/2001/XMLSchema#in
9	KeyError	"71"^^<http://www.w3.org/2001/XMLSchema#in
10	CalledProcessError	"68"^^<http://www.w3.org/2001/XMLSchema#in

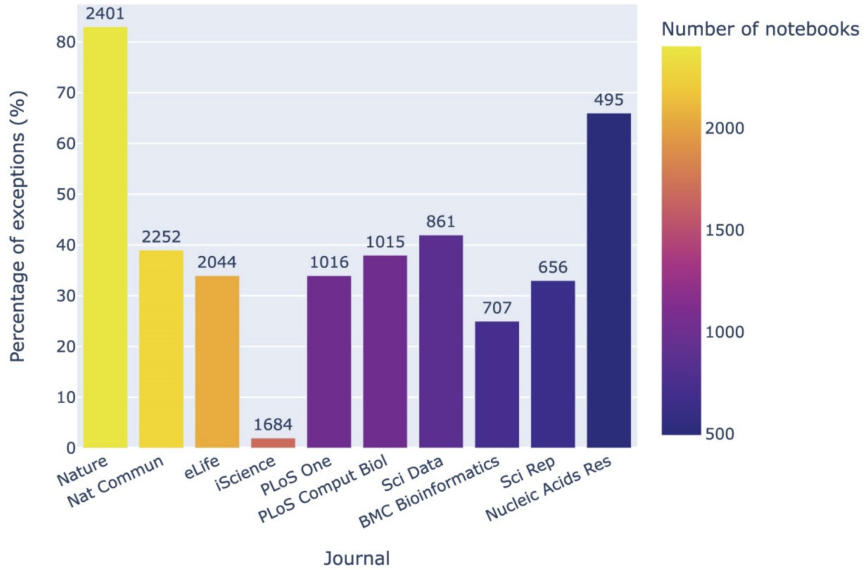
Showing 1 to 10 of 10 entries



SPARQL Query: Exceptions occurring in Jupyter notebooks in our corpus.

Results - Notebook Reproducibility





SPARQL Query: [Jupyter notebook exceptions by research field, taking as a proxy the highest-level MeSH terms of the article associated with the notebook.]

Effect of peer review and editorial workflows: Nature has about 50 *percent* more notebooks than iScience, but Nature's fail about 40 *times* more often

Example queries

- Local
  - Notebooks by search term: ‘immun’ AND (‘stem’ OR ‘differentiation’)
  - Article by keywords, e.g., ‘open source’
  - Most common errors in immunology
  - Most common errors in Nature journal
  - MeSH terms ranked by ‘ModuleNotFoundError’ frequency
  - GitHub repositories
    - GitHub repositories by their stargazers count
    - GitHub repositories and their Software Heritage snapshot
      - https://doi.org/10.5281/zenodo.12806151
  - Articles and repositories with notebooks in Julia
- Federated
  - Match articles between FAIR Jupyter and Wikidata via DOI
  - Match articles between FAIR Jupyter and NFDI/MaRDI via DOI and get co-used software

Impact of FAIR Jupyter on Research Software Engineering

- Enhancing Reproducibility:
  - Provides insights into the computational reproducibility of Jupyter notebooks in research software.
- Metadata-Driven Analysis:
  - Captures dependencies, execution environments, and versioning to improve software reliability.
- Knowledge Graph Integration:
  - Enables structured exploration of software repositories, aiding in dependency management and best practices.

Impact of FAIR Jupyter on Research Software Engineering

- FAIR Principles Applied:
  - Findability: Links software, notebooks, and publications for better discoverability.
  - Accessibility: Public SPARQL endpoint allows open data exploration.
  - Interoperability: Supports integration with external resources and standards.
  - Reusability: Encourages best practices in research software development.
- Driving Open Science & Software Quality:
  - Identifies trends in dependency usage and failures.
  - Supports sustainable and transparent software engineering practices.

Future Work: FAIR Jupyter for SE & Education

- Educational Applications
  - Use FAIR Jupyter to teach best practices in reproducible research and software development.
  - Enable hands-on learning with real-world datasets and reproducibility challenges.
- Enhancing Software Quality & Standards
  - Analyze trends in software dependencies, versioning, and execution environments.
  - Develop guidelines for improving reproducibility in research software.
- Expanding the Knowledge Graph
  - Link FAIR Jupyter with other open-source datasets to provide a broader view of software engineering practices.
  - NFDIs: FAIR Jupyter / KGI4NFDI / nfdi.software / …
- Community Engagement & Collaboration
  - Foster open science initiatives by engaging researchers, educators, and developers.
  - Encourage contributions to enhance FAIR Jupyter’s utility in software engineering.

Thank you

- German Research Foundation (DFG), TRR-386, project number 514664767
- Alfred P. Sloan Foundation under grant number G-2021-17106.
- Ara Cluster of Friedrich Schiller University Jena under DFG grants INST 275/334-1 FUGG and INST 275/363-1 FUGG.
- BASE4NFDI/KGI4NFDI: DFG 521453681

- **Contact**: sheeba.samuel@informatik.tu-chemnitz.de & daniel.mietchen@fiz-karlsruhe.de