

Introduction and tutorial on Metadata and SMPs for FAIR research software

Enabling bridges across research artifacts



Leyla Jael Castro
Semantic Technologies Team, ZB MED
NFDI4DataScience

[DOI:10.5281/zenodo.13799879](https://doi.org/10.5281/zenodo.13799879)



DFG
Deutsche
Forschungsgemeinschaft



Name: Introduction and tutorial on Metadata and SMPs for FAIR research software

Description: This talk/tutorial is organized in four parts: Introduction to ELIXIR Software Management Plans, Metadata for Research Software, FAIR for Research Software, and machine-actionable Software Management Plans. Hands-on are included for part 2 (Metadata for Research Software) and part 3 (FAIR for Research Software). There is an optional part that will briefly introduce JSON/JSON-LD if needed. This talk/tutorial reuses training materials used in other tutorials, they are indicated in the slides as a link to the original material.

Keywords: Research software, metadata, FAIR₄RS



Questions

- Part 1 - Introduction to ELIXIR Software Management Plans
 - What is an SMP?
 - What are the benefits of SMPs?
- Part 2 - Metadata for Research Software
 - What is metadata and what are its benefits?
 - What metadata efforts exist for research software?
 - How can I use Bioschemas markup in my GitHub pages?
- Part 3 - FAIR for Research Software
 - What are the FAIR₄RS principles?
 - What practical steps can I take to implement the FAIR₄RS?
 - How are those steps translated into metadata?
- Part 4 - maSMPs
 - What are maSMPs?
 - What is the metadata considered in maSMPs?



Learning Outcomes

- Describe what SMPs are and what maSMPs add to them
- Describe what metadata is and what options exist for research software
- Use schema.org and Bioschemas profiles on GitHub pages
- Identify practical good practices towards FAIR₄RS



Requirements

- Familiarity on how to use GitHub
- Basic knowledge on how to use GitHub pages. More information at [GitHub Pages](#)
- Familiarity with JSON-LD
- Basic knowledge of Python (and having it install)
- Basic knowledge of Docker (and having it install)

Additional info



Time estimation 60 minutes (90 including the GitHub pages tutorial)



Level Beginner / Introductory



Published 2024-09-20



Latest modification 2024-09-20



License CC-By 4.0



Version 1.0.0



Identifier DOI:10.5281/zenodo.



Citation Castro, LJ. (2024, September 20). Adding Bioschemas Dataset and ComputationalTool markup to GitHub pages. Zenodo. <https://doi.org/10.5281/zenodo>.

A core element of Open Science is the use of Data Management Plans. They describe the data management life cycle for the data to be collected, processed and/or generated within the lifetime of a particular project or activity. A Software Management Plan (SMP) plays the same role but for software.



ELIXIR SMPs

Improving the **quality** and **sustainability** of **research software** by producing, **adopting**, **promoting** and **measuring** information standards and best practices applied to software development life cycle.

- José María Fernández (ES)
- Dimitrios Bampalakis (SE)
- Leyla Jael Castro (DE)
- Renato Alves (DE)
- Eva Martin del Pico (ES)
- Fotis Psomopoulos (GR)
- Shoaib Sufi (UK)
- Allegra Via (IT)

Towards a unified approach to software development and management in ELIXIR

- Similarly to Data Management Plans, SMP is an **awareness** tool.
- Think in advance about the software that will be developed
- Think about most important parts of software project
- Think about roles and responsibilities in software project
- Use it as a guide for everyone involved in the project

Design Principles behind this SMP

- Focused on Life Sciences
- Minimal set of questions
- Applicable to even very small software projects
- In the context of FAIR
- First step / a pilot, with the goal of raising awareness within the ELIXIR community and collecting two rounds of feedback that was incorporated in the final version

- PI submitting project proposals
 - thinking ahead about software development and management practices
- Project Manager, checking the development process
 - is there a license?
 - is project available online?
- Research Software Engineer / Coding Researcher
 - use SMP as a playbook
 - use SMP as a checklist

Metadata help us describe things:
publications, data, software

Metadata enabling bridges but also
*ilities

Structured metadata plays a key role in
FAIR,
facilitates generation of KGs,
and makes things easier for machines and
humans

Big community effort, cultural change

Metadata for Research Software

Why do we need metadata?



As open as possible as close as necessary

But... we still need some minimum information on that that is closed
(same on that that is open)

Metadata help us describe things: publications, data, software

Metadata enabling bridges but also *ilities

Structured metadata plays a key role in **FAIR**,
facilitates generation of **KGs**,
and makes things easier for machines and humans



You use (and benefit) from metadata on regular basis

Ice-breaking exercise

- ▶ Ask the person next to you to describe something, e.g., research software, movie, dataset, person. Be careful that no “name” is used
- ▶ Can you guess what the person is describing?
- ▶ If you try the description in a search engine or GPT, can they guess it?

- ▶ Share your experiences with the group
- ▶ What metadata was used?
- ▶ What metadata was key?

How does metadata look like?

PubMed®

knowledge graphs

Search

Advanced Create alert Create RSS User Guide

Save Email Send to

Sort by: Best match

Display options



OpenAlex

Works

Search OpenAlex

Log in Sign up

Unserved search

Show works where:

Work is Knowledge graphs and their applications in... X

Works

Knowledge graphs and their applications in drug discovery

2021 Finlay MacLean Expert Opinion on Drug Discovery

Cited by 45

1 results

open access

topic

institution

year

2021

type

review

0%

Analysis of Gene Interaction Networks

Benevolent (Kingdom)

Export results

Spreadsheet (.csv)

Shorten column values for Excel compatibility?

Endnote format (.ris)

WoS format (.txt)

Cancel Start export

```
<PubmedArticleSet>
  <PubmedArticle>
    <MedlineCitation Status="MEDLINE" Owner="NLM">
      <PMID Version="1">33843398</PMID>
      <DateCompleted>
        <Year>2021</Year>
        <Month>09</Month>
        <Day>20</Day>
      </DateCompleted>
      <DateRevised>
        <Year>2022</Year>
        <Month>12</Month>
        <Day>07</Day>
      </DateRevised>
    </MedlineCitation>
    <Article PubModel="Print-Electronic">
      <Journal>
        <ISSN IssnType="Electronic">1746-045X</ISSN>
        <JournalIssue CitedMedium="Internet">
          <Volume>16</Volume>
          <Issue>9</Issue>
          <PubDate>
            <Year>2021</Year>
            <Month>Sep</Month>
          </PubDate>
        </JournalIssue>
        <Title>Expert opinion on drug discovery</Title>
        <ISOAbbreviation>Expert Opin Drug Discov</ISOAbbreviation>
      </Journal>
      <ArticleTitle>Knowledge graphs and their applications in drug discovery.</ArticleTitle>
      <Pageitation>
        <StartPage>1057</StartPage>
        <EndPage>1069</EndPage>
        <MedlinePgn>1057-1069</MedlinePgn>
      </Pageitation>
      <ElocationID EIdType="doi" ValidYN="Y">10.1080/17460441.2021.1910673</ElocationID>
    </Article>
    <Abstract>
      <AbstractText Label="INTRODUCTION">Knowledge graphs have proven to be promising system toward a systems biology approach, Knowledge graphs have emerged as attractive methods <AbstractText Label="AREAS COVERED">In this review, the author summarizes the applicat insights, highlighting target identification and drug repurposing as two areas showing the dangers of degree and literature bias, and discuss mitigation strategies.</AbstractText Label="EXPERT OPINION">Whilst knowledge graphs and graph-based machine l data, and only highlight biological associations, failing to model causal relationship</Abstract>
    </Abstract>
    <AuthorList CompleteYN="Y">
      <Author ValidYN="Y">
        <LastName>MacLean</LastName>
        <ForeName>Finlay</ForeName>
        <Initials>F</Initials>
        <Identifier Source="ORCID">0000-0003-2779-179X</Identifier>
      </Author>
    </AuthorList>
  </PubmedArticle>
</PubmedArticleSet>
```

How does metadata look like?

ZB MED Homepage > Footermenu > Contact details > Leyla Jael Castro

Leyla Jael Castro

Team Leader Semantic Technologies

Key tasks and activities

- ▶ Literature-based information retrieval
- ▶ Recommendation systems
- ▶ Ontology-based search and categorization
- ▶ Interdisciplinary research in the Life Sciences domain

Profile

"I am a Computer Scientist interested in semantic web, linked data, data science, open science and education. I am currently involved in community projects aiming to make FAIR a reality not only for data but also for software and training materials. I have worked on software development and data integration semantic, project coordination, scientific events organization and chairing, and community-based projects (e.g. Bioschemas and BioJS). I have also worked as a university lecturer on software development and information systems."

Educational background

- ▶ 2012 – 2017
PhD Computer Languages and Systems (with a focus on AI), Universitat Jaume I, Spain
- ▶ 2006 – 2007
Graduated diploma in education, Universidad Militar Nueva Granada, Colombia
- ▶ 2004 – 2006
M. Sc. Computer Science, Universidad de los Andes, Colombia
- ▶ 1992 – 1997
Bachelor Computer Science, Universidad de los Andes, Colombia

Schema.org

familyName	Castro
givenName	Leyla Jael
affiliation	<ul style="list-style-type: none">- ZB MED- NFDI4DataScience
knowsAbout	Metadata for research artifacts, occasionally domain-specific metadata (life sciences related), semantic web, data science and artificial intelligence, semantics + AI
identifier	ORCID:0000-0003-3986-0510
mainEntityofPage	https://www.zbmed.de/en/contact-details/leyla-jael-castro
alumniOf	<ul style="list-style-type: none">- University of the Andes (Colombia)- Military University Nueva Granada (Colombia)- University Jaume I (Spain)

```
{
  "@context": "http://schema.org",
  "@type": "Person",
  "@id": "https://orcid.org/0000-0003-3986-0510",
  "familyName": "Castro",
  "givenName": "Leyla Jael",
  "jobTitle": "Team leader"
}
```



Structured metadata

- ▶ Structured representation
- ▶ Local purpose
- ▶ Examples
 - Databases - e.g., relational model
 - JSON - JSON schema
 - XML - XSD schema

Semantically structured metadata

- ▶ Structured representation encapsulating “meaning”
 - Logics
 - Inference rules
- ▶ Global purpose
- ▶ Statements = Triplets
 - domain - relation - range
 - class - property - class/primitive
 - type - property - type/primitive
 - subject - predicate - object

Types:

Close hierarchy / Open hierarchy

Thing -

- ▶ Action +
- ▶ BioChemEntity +
- ▶ CreativeWork +
- ▶ Event +
- ▶ Intangible +
- ▶ MedicalEntity +
- ▶ Organization +
- ▶ Person +
- ▶ Place +
- ▶ Product +
- Taxon

CreativeWork

A Schema.org Type

Thing - [CreativeWork](#)

The most generic kind of creative work, including books, movies, photographs, software programs, etc.

[more...]

Property	Expected Type	Description
Properties from CreativeWork		
about	Thing	The subject matter of the content. <i>Owner property, will be RDF</i>
abstract	Text	An abstract or short description that summarizes a CreativeWork.
accessMode	Text	The human sensory perceptual system or cognitive faculty through which a person may process or perceive information. Values should be drawn from the approved vocabulary.
accessModeSufficient	Text, List	A set of single or combined accessModes that are sufficient to understand all the intellectual content of a resource. Values should be drawn from the approved vocabulary.
accessibilityAPI	Text	Indicates that the resource is compatible with the referenced accessibility API. Values should be drawn from the approved vocabulary.
accessibilityControl	Text	Identifies input methods that are sufficient to fully control the described resource. Values should be drawn from the approved vocabulary.
accessibilityFeature	Text	Content features of the resource, such as accessible media, alternative and supported enhancements for accessibility. Values should be drawn from the approved vocabulary.
accessibilityHazard	Text	A characteristic of the described resource that is physiologically dangerous to some users. Related to WCAG 2.0 guideline 2.2. Values should be drawn from the approved vocabulary.
accessibilitySummary	Text	A human readable summary of specific accessibility features or deficiencies, consistent with the other accessibility metadata but expressing subtleties such as "short descriptions are present but long descriptions will be needed for non-visual users" or "short descriptions are present and no long descriptions are needed".
accountablePerson	Person	Specifies the Person that is legally accountable for the CreativeWork.
actualScenePage	CreativeWork or URL	Indicates a page documenting how scenes can be purchased or otherwise acquired, for the current item.
aggregating	AggregatingPage	The overall thing, based on a collection of reviews or ratings of the item.
alternateIssue	Text	A secondary title of the CreativeWork.
articleDate	URL or WebPage	Indicates single or other "in motion" articles of the CreativeWork. In the case of MediaObject, the term MediaObject has been used to indicate that the content is accessible, but is archived by archive.org, journalist, activist, or law enforcement organizations. In such cases, the referenced page may not directly publish the content.
assesses	DefinitionTerm or Text	The item being described is intended to assess the competency or learning outcome defined by the referenced item.
associatedMedia	MediaObject	A media object that encodes the CreativeWork. This property is a synonym for encoding.
audience	Audience	An intended audience, i.e. a group for whom something was created. Supercedes terms like reader.
audio	AudioObject or Clip or AudioRecording	An embedded audio object.
author	Organization or Person	The author of the content or rating. Please note that author is special in that HTML. It provides a specific mechanism for indicating authorship via the rel tag. That is equivalent to this and may be used interchangeably.
award	Text	Award won by or for this item. Supercedes awards.
character	Person	Fictional person connected with a creative work.
citation	CreativeWork or Text	A citation or reference to another creative work, such as another publication, web page, scholarly article, etc.
comment	Comment	Comments, typically from users.
commentCount	Integer	The number of comments the CreativeWork (e.g. Article, Question or Answer) has received. This is most applicable to works published in Web sites with commenting system; additional comments may exist elsewhere.
contentLocation	Text	Conditions that affect the availability of, or method(s) of access to, an item. Typically used for new work items such as an ArticleComponent that has an ArticleComponent. This property is not suitable for use as a general Web access control mechanism. It is expressed only in natural language. For example: "Available by appointment from the Reading Room" or "Accessible only from logged-in accounts".
contentLocation	Place	The location depicted or described in the content. For example, the location in a photograph or painting.
contentRating	Rating or Text	Official rating of a piece of content—for example, MPAA PG-13.
contentReferenceTime	DateTime	The specific time described by a creative work, for works (e.g. articles, video objects etc.) that emphasize a particular moment within a CreativeWork.
contributor	Organization or Person	A secondary contributor to the CreativeWork or Event.

CodeMeta

Terms from Schema.org

Recognized properties for CodeMeta [SoftwareSourceCode](https://schema.org/SoftwareSourceCode) and [SoftwareApplication](https://schema.org/SoftwareApplication) includes the following terms from <https://schema.org>. These terms are part of the CodeMeta specification and can be used without any prefix.





Property	Type	Versions	Description
<code>applicationCategory</code>	Text or URL	v2, v3	Type of software application, e.g. 'Game, Multimedia'.
<code>applicationSubCategory</code>	Text or URL	v2, v3	Subcategory of the application, e.g. 'Arcade Game'.
<code>author</code>	Organization or Person	v2, v3	The author of this content or rating. Please note that author is special in that HTML 5 provides a special mechanism for indicating authorship via the rel tag. That is equivalent to this and may be used interchangeably.
<code>citation</code>	CreativeWork or URL	v2, v3	A citation or reference to another creative work, such as another publication, web page, scholarly article, etc.
<code>codeRepository</code>	URL	v2, v3	Link to the repository where the un-compiled, human readable code and related code is located (SVN, GitHub, CodePlex, institutional GitLab instance, etc.).

Codemeta terms

The CodeMeta project also introduces the following additional properties, which lack clear equivalents in <https://schema.org> but can play an important role in software metadata records covered by the CodeMeta crosswalk.

Property	Type	Versions	Description
<code>buildInstructions</code>	URL	v2, v3	link to installation instructions/documentation
<code>continuousIntegration</code>	URL	v2	link to continuous integration service
<code>developmentStatus</code>	Text	v2, v3	Description of development status, e.g. Active, inactive, suspended. See repostatus.org



Property	Expected Type	Description	CD	Controlled Vocabulary	Example
Marginality: Minimum.					
description	Text	Schema: A description of the item. Bioschemas: A short description of the tool.	ONE		
name	Text	Schema: The name of the item.			
url	URL	Schema: URL of the item. Bioschemas: Homepage of the tool.	ONE		
Marginality: Recommended.					
applicationCategory	Text URL	Schema: Type of software application, e.g. 'Game, Multimedia'. Bioschemas: Type of tool e.g. Command-line tool, Web application etc. Note:	MANY	Please use terms from the 'Tool type' table in the biotools documentation	

About

BioHackathon project to define and follow up BioHackathon projects

biohackeu20

- Readme
- Apache-2.0 license
- Cite this repository -
- Activity
- Custom properties
- 1 star
- 7 watching
- 0 forks
- Report repository

Contributors 2



ljgarcia



grglazarov Georgi Lazarov

Languages



Additional details

Related works

Is derived from

<https://github.com/zbmed-semtec/TREC-doc-2-doc-relevance/tree/v1.0.0> (URL)

Is supplemented by

<https://zenodo.org/record/7338056> (URL)

<https://zenodo.org/record/7324822> (URL)

References

https://trec.nist.gov/data/t14_genomics.html (URL)

<https://trec.nist.gov/pubs/trec14/papers/GEO.OVERVIEW.pdf> (URL)

References

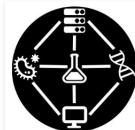
- Giraldo O, Solanki D, Rebholz-Schuhmann D, Castro LJ. Fleiss kappa for doc-2-doc relevance assessment. Zenodo; 2022. doi:10.5281/zenodo.7338056
- Giraldo O, Solanki D, Rebholz-Schuhmann D, Castro LJ. Fleiss kappa for doc-2-doc relevance assessment. Zenodo; 2022. doi:10.5281/zenodo.7338056
- Hersh W, Cohen A, Yang J, Bhupatiraju RT, Roberts P, Hearst M. TREC 2005 Genomics Track Overview. : 26.

Schema.org metadata in action: GitHub pages

BioHackOutcomes

▲ Get JSON-LD 1 → Visit SoftwareSourceCode

name	BioHackOutcomes
description	BioHackathon project to define and follow up BioHackathon projects.
url	https://github.com/zbmec-semtec/BioHackOutcomes
author	<div>→ Visit Person Object</div> <div><div>familyName: Lazarov</div><div>givenName: Georgi</div></div> <div>→ Visit Person Object</div> <div><div>familyName: Castro</div><div>givenName: Leyla Jael</div></div>
license	<div>→ Visit CreativeWork Object</div> <div><div>name: APACHE LICENSE, VERSION 2.0</div><div>url: https://www.apache.org/licenses/LICENSE-2.0</div></div>



Lecture on Biomedical semantics, information retrieval and knowledge discovery

Motivation

- This lecture on "Biomedical semantics, information retrieval and knowledge discovery" aims at introducing studies with no previous knowledge on the topics to the world of the Semantic Web and Linked Data with a special focus on Biomedical knowledge and resources.
- You will learn how to use data in a way that it is human and machine readable and processable.
- Semantics is used to add labels and meaning to data and to offer opportunities to gather and analyse data around labels, meaning and human understandable topics.
- The introduction of semantics requires special formats and standards to make the data useful for the general public to provide interoperability.

Who can use this material?

By now, only students enrolled in a course using these training materials will have access to the full content.

License: This material is distributed under CC BY-NC.

Projects metadata

OntoClue

▲ Get JSON-LD

▲ Get RO Crate

■ RO-Crate HTML Preview

Started in 2021-01-01

Description

OntoClue aims to provide a framework to optimize and compare document-similarity and doc2doc-relevance approaches based on word-embeddings and document-embeddings. Using the RELISH dataset, each approach creates document-embeddings and calculates the Cosine Similarity. An optimizer finds the best hyperparameter combination that naturally (i.e., with no further tuning or training) resembles better the three document relevance assessments coming from RELISH. The approaches are compared using Precision (P@N) and Normalized Discounted Cumulative Gain (NDCG@N). TREC 2005 Genomics Track data has also been analyzed using a repurposed version that transforms document-to-topic relevance into document-to-document relevance. The main focus of this project relies on RELISH.

Table of contents

OntoClue
Description
Keywords
Url
Current project members
Previous project members
Department
Parent organization, consortium or research project
Funding
Outcomes
External contributors

RO-crate for the research project 'OntoClue'

Download all the metadata for RO-crate for the research project 'OntoClue' in JSON-LD format
[Check this crate](#)

RO-crate for the research project 'OntoClue'

@id	/
name [?]	RO-crate for the research project 'OntoClue'
@type	Dataset
description [?]	This RO-crate describes the research project 'OntoClue' including metadata about it created as part of it. The outcomes are listed via the hasPart property of the RO-crate view of the research outcomes.
datePublished [?]	2023-11-02
license [?]	http://spdx.org/licenses/CC-BY-4.0
keywords [?]	<ul style="list-style-type: none">machine-actionabilitysoftware management plansresearch softwarecontrolled vocabularymetadata schemaongoingSMP

```
{
  "@context": "http://schema.org/",
  "@type": "SoftwareSourceCode",
  "@id": "https://github.com/zbmec-semtec/BioHackOutcomes",

  "name": "BioHackOutcomes",
  "description": "BioHackathon project to define and follow up BioHack",
  "url": "https://github.com/zbmec-semtec/BioHackOutcomes",

  "author": [
    {
      "@type": "Person",
      "@id": "https://orcid.org/0000-0002-0762-4305",
      "familyName": "Lazarov",
      "givenName": "Georgi"
    },
    {
      "@type": "Person",
      "@id": "https://orcid.org/0000-0003-3986-0510",
      "familyName": "Castro",
      "givenName": "Leyla Jael"
    }
  ],
  "license": {
    "@type": "CreativeWork",
    "@id": "http://spdx.org/licenses/Apache-2.0",
    "name": "APACHE LICENSE, VERSION 2.0",
    "url": "https://www.apache.org/licenses/LICENSE-2.0"
  },
  "citation": [
    "Georgi L, Castro LJ. BioHackathon Outcomes GitHub Metadata. GitHub",
    "Castro LJ, Martin C, Lazarov G, Cernokova D, Takatsuki T, Harrow"
  ],
  "codeRepository": "https://github.com/zbmec-semtec/BioHackOutcomes",
  "programmingLanguage": [
    "Python"
  ],
  "keywords": [
    "BioHackathon Europe 2020", "GitHub", "Metrics"
  ]
}
```

Let's hear it from you!

<https://forms.gle/gySq41Pnc54NpvRi8>

- ▶ Thinking on the research software metadata summary
 - How is this useful to you?
 - What are the advantages of such a summary?
 - Would you like to have a similar improved summary automatically created for your software?

- ▶ Thinking on the research software metadata in JSON-LD
 - What uses does it have?
 - What are the advantage of this format?

- ▶ Disclaimer
 - Participation is voluntary
 - We will analyze responses to understand better researchers needs wrt research software metadata
 - We are not collecting any personal or identifiable information

The following slides are an excerpt from a tutorial presented at the FDO FDO Summit 2024 in Berlin

- Tutorial name: Practical web-based FDOs with RO-Crate and FAIR Signposting
- This tutorial was presented at <https://fairdo.org/fdof-summit-2024/>
- Material available at <https://drive.google.com/file/d/1TKxFpz4QcG9bGCrGOB8Zq9kZU-H0kc6S/view>
- If you reuse this part, please cite the published version as

Soiland-Reyes S. A very brief introduction to making metadata with JSON-LD [version 1; not peer reviewed]. F1000Research 2023, 12(ELIXIR):619 (slides) (<https://doi.org/10.7490/f1000research.1119460.1>)

A very brief introduction to making metadata with JSON-LD

Stian Soiland-Reyes

The University of Manchester

RO-Crate co-lead

soiland-reyes@manchester.ac.uk

<https://orcid.org/0000-0001-9842-9718>



The University of Manchester

Leyla Jael Castro

ZB MED Information Centre for Life Sciences

Bioschemas chair, NFDI4DataScience member

ljgarcia@zbmed.de

<https://orcid.org/0000-0003-3986-0510>



This work is licensed under a
[Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

JSON for Linking Data

Data is messy and disconnected. JSON-LD organizes and connects it, creating a better Web.

<https://json-ld.org>

⇔ Linked Data

Linked Data empowers people that publish and use information on the Web. It is a way to create a network of standards-based, machine-readable data across Web sites. It allows an application to start at one piece of Linked Data, and follow embedded links to other pieces of Linked Data that are hosted on different sites across the Web.

A Simple Example

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "@id": "http://dbpedia.org/resource/John_Lennon",
  "name": "John Lennon",
  "born": "1940-10-09",
  "spouse": "http://dbpedia.org/resource/Cynthia_Lennon"
}
```

{ JSON-LD

JSON-LD is a lightweight Linked Data format. It is easy for humans to read and write. It is based on the already successful JSON format and provides a way to help JSON data interoperate at Web-scale. JSON-LD is an ideal data format for programming environments, REST Web services, and unstructured databases such as CouchDB and MongoDB.

<https://www.w3.org/TR/json-ld>

A word (or two) about JSON/JSON-LD

```
{ "@context": "https://schema.org/",
```

JSON-LD **context**

```
"@type": "LearningResource",
```

Type (identifier implied)

```
"http://purl.org/dc/terms/conformsTo": {
```

```
  "@type": "CreativeWork",
```

Bioschemas **profile**

```
  "@id": "https://bioschemas.org/profiles/TrainingMaterial/1.0-RELEASE"
```

```
},
```

```
  "name": "Adding nanomaterial data",
```

```
  "version": "0.9.3",
```

```
  "description": "This tutorial describes how nanomaterial data can  
be added to an eNanoMapper server using a RDF format.",
```

Metadata about training
material

```
  "license": "https://creativecommons.org/licenses/by/4.0/",
```

```
  "keywords": "ontologies, enanomapper, RDF",
```

```
  "url": "https://nanocommons.github.io/tutorials/enteringData/",
```

```
  "provider": {
```

```
    "@type": "Organization",
```

```
    "name": "NanoCommons",
```

```
    "url": "https://www.nanocommons.eu/"
```

```
  },
```

Nested object

with its own @type and
attributes

```
  "...": {}
```

<https://nanocommons.github.io/tutorials/enteringData/>
by Egon Willighagen

A word (or two) about JSON/JSON-LD

```
{
  "@id": "figure.png",
  "@type": ["File", "ImageObject"],
  "name": "XXL-CT-scan of an XXL Tyrannosaurus rex skull",
  "identifier": "https://doi.org/10.5281/zenodo.3479743",
  "author": {"@id": "https://orcid.org/0000-0002-8367-6908"},
  "encodingFormat": "image/png"
}
```

Metadata

```
{
  "@id": "https://orcid.org/0000-0002-8367-6908",
  "@type": "Person",
  "affiliation": { "@id": "https://ror.org/03f0f6041" },
  "name": "J. Xuan"
}
```

```
{
  "@id": "https://ror.org/03f0f6041",
  "@type": "Organization",
  "name": "University of Technology Sydney",
  "url": "https://www.uts.edu.au/"
}
```

Linked Data: Reference by URI

Types and properties are expanded by context, e.g. <http://schema.org/ImageObject>

Entities can be **cross-referenced** with `@id` within the same JSON-LD document

Style *Flattened JSON-LD*: each entity is listed separately in `@graph` array. `@id` required

Style *Compacted JSON-LD*: the entity can be nested within any cross-reference, `@id` optional

Clients can still follow the links for potentially more data (e.g. ORCID lists publications)

A word (or two) about JSON/JSON-LD

JSON-LD Playground

Play around with JSON-LD markup by typing out some JSON below and seeing what gets generated from it at the bottom of the page. Pick any of the examples below to get started.

NOTES:

- The playground uses `jsonld.js` which conforms to [JSON-LD 1.1 syntax \(errata\)](#), [API \(errata\)](#), and [framing \(errata\)](#).
- Other related playgrounds: [Classic JSON-LD 1.0 Playground](#) | [RDF Distiller](#) | [CHAPI Playground](#)

Examples: Person Event Place Product Recipe

JSON-LD Input Options

```
{
  "@context": "https://schema.org/",
  "@type": "LearningResource",
  "http://purl.org/dc/terms/conformsTo": {
    "@type": "CreativeWork",
    "@id": "https://bioschemas.org/profiles/TrainingMaterial",
    "name": "Adding nanomaterial data",
    "version": "0.9.3",
    "description": "This tutorial describes how nanomaterials are added to a database.",
    "license": "https://creativecommons.org/licenses/by/4.0",
    "keywords": "ontologies, enanomap, RD",
    "url": "https://nanocommons.github.io/tutorials/enterin",
    "provider": {
      "@type": "Organization",
      "name": "NanoCommons",
      "url": "https://www.nanocommons.eu/"
    }
  }
}
```

Testing in the JSON-LD Playground

5525169

- @type: LearningResource
- http://purl.org/dc/terms/conformsTo
- name: Adding nanomaterial data
- version: 0.9.3
- description: This tutorial describes how...
- license: https://creativecommons.org...
- keywords: ontologies, enanomap, RD...
- url: https://nanocommons.github...
- provider

<https://json-ld.org/playground/>



E1. A GitHub page for your research software

February 7, 2024 (1.0.0)

Lesson

Open

Adding Bioschemas Dataset and ComputationalTool markup to GitHub pages

Castro, Leyla Jael 

bioschemas-ghpages-markup-tutorial Overview Name: Tutorial on adding Bioschemas markup to GitHub pages Description: This tutorial shows how to add Bioschemas markup to GitHub pages. It uses a simple GitHub page hosted in the gh-pages branch to create a sample project page, i.e., as learners could do with their own GitHub projects. As an exa...

Part of [Semantic Technologies team at ZB MED Information Centre for Life Sciences](#), [NFDI for Data Science and AI](#)

Uploaded on February 7, 2024

 48

 42

<https://doi.org/10.5281/zenodo.10629452>

Please use the latest version

If you use this part please cite it as

Castro, L. J. (2024, February 7). Adding Bioschemas Dataset and ComputationalTool markup to GitHub pages. Zenodo. <https://doi.org/10.5281/zenodo.10629453>

Metadata help us describe things:
publications, data, software

Metadata enabling bridges but also
*ilities

Structured metadata plays a key role in
FAIR,
facilitates generation of KGs,
and makes things easier for machines and
humans

Big community effort, cultural change



FAIR for Research Software

Findable: Software, and its associated metadata, is easy for both humans and machines to find

(=) F1 Software is assigned a globally unique and persistent identifier

(new) F1.1 Components of the software representing levels of granularity are assigned distinct identifiers

(new) F1.2 Different versions of the software are assigned distinct identifiers

(=) F2 Software is described with rich metadata

(=) F3 Metadata clearly and explicitly include the identifier of the software they describe

(=) F4 Metadata are FAIR, searchable and indexable

Accessible: Software, and its metadata, is retrievable via standardised protocols

(=) A1 Software is retrievable by its identifier using a standardised communications protocol

(=) A1.1 The protocol is open, free, and universally implementable

(=) A1.2 The protocol allows for an authentication and authorization procedure, where necessary

(=) A2 Metadata are accessible, even when the software is no longer available

Interoperable: Software interoperates with other software by exchanging data and/or metadata, and/or through interaction via application programming interfaces (APIs), described through standards

(≠) I1. Software reads, writes and exchanges data in a way that meets domain-relevant community standards

~~I2. (meta)data use vocabularies that follow FAIR principles~~

(=) I2(3). Software includes qualified references to other objects

Reusable: Software is both usable (can be executed) and reusable (can be understood, modified, built upon, or incorporated into other software)

(=) R1. Software is described with a plurality of accurate and relevant attributes

(=) R1.1. Software is given a clear and accessible license

(=) R1.2. Software is associated with detailed provenance

(new) R2 Software includes qualified references to other software

(=) R3(1.3) Software meets domain-relevant community standards

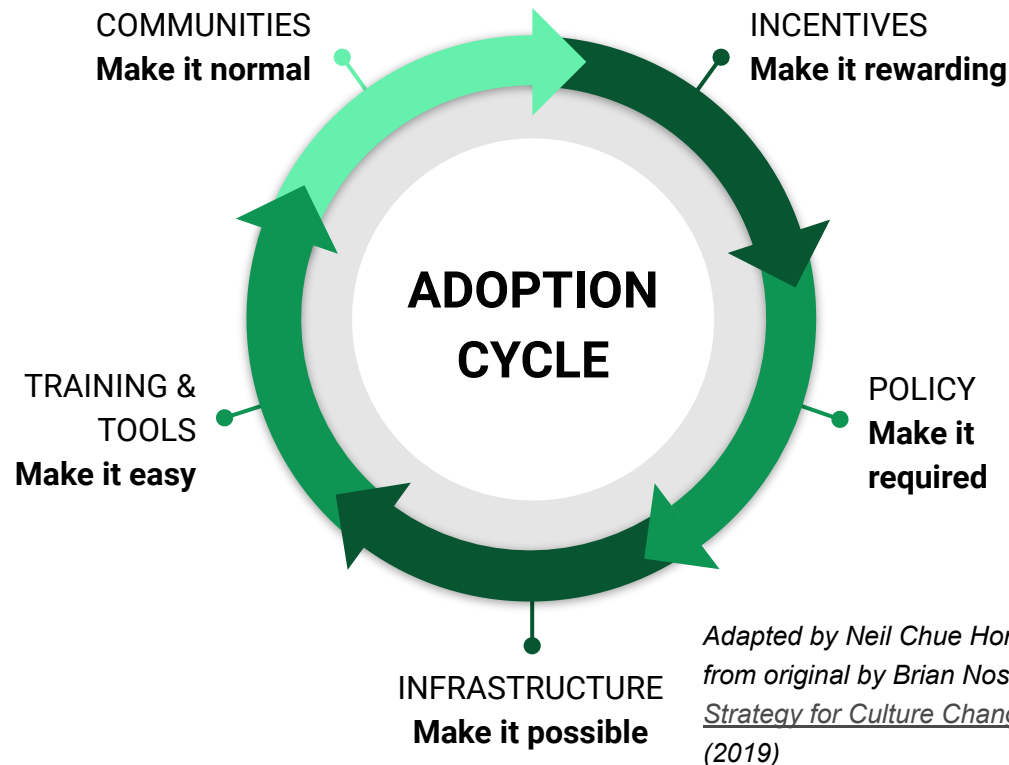
Who is responsible for FAIR software?

Who is expected to apply FAIR?

► And why?

“...the application of the FAIR4RS Principles is the responsibility of the owners (who are often the creators) of the software, not the users. “

“The FAIR4RS Principles are also relevant to the larger ecosystem and various stakeholders that support research software (e.g., repositories and registries).”



Moving towards FAIR4RS

Archives/Repositories

F1 Software is assigned a globally unique and persistent identifier

A1 Software is retrievable by its identifier using a standardised communications protocol

A1.1 The protocol is open, free, and universally implementable

F1.2 Different versions of the software are assigned distinct identifiers

<https://choosealicense.com>

ORCID

Citation File Format (CFF)

zenodo



Software Heritage

v3.12.4



R2 Software includes qualified references to other software

Registries

F4 Metadata are FAIR, searchable and indexable

A2 metadata are accessible, even when the software is no longer available

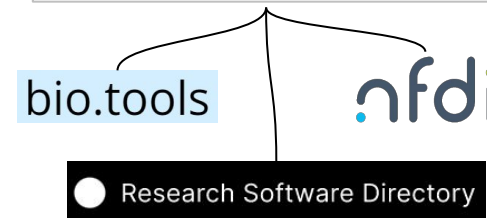
F3 Metadata clearly and explicitly include the identifier of the software they describe

Archives/Repositories/Registries

R1.1. Software is given a clear and accessible license

R1.2. Software is associated with detailed provenance

I2. Software includes qualified references to other objects



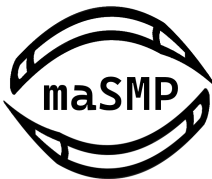
F2 Software is **described** with rich metadata

R1. Software is **described** with a plurality of accurate and relevant attributes



Schema.org

CodeMeta



FAIR4ML



E1. Adding a missing license to your public repo

No license no reuse

1. Go to any of your own public GitHub repos with no license
2. Navigate to Insights and then to Community Standards
3. Select Add on the license row
4. You will see the license templates
5. Choose the one you want
6. Commit changes and now you have a license!



Step by step in the following slides

E1. Adding a missing license to your public repo

The screenshot shows the GitHub Insights interface. At the top, there are tabs for Projects, Security, Insights, and Settings. An orange arrow labeled '1.' points to the 'Insights' tab. Below the tabs, there is a sidebar menu with options: Pulse, Contributors, Community, Community Standards, Traffic, Commits, Code frequency, Dependency graph, Network, Forks, and People. An orange arrow labeled '2.' points to the 'Community Standards' option. The main content area is titled 'Community Standards' and contains a checklist of various standards. An orange arrow labeled '3.' points to the 'License' row in the checklist, which has an 'Add' button next to it. The 'License' row also includes a link 'Choosing a license'.

1. Go to any of your own public GitHub repos with no license

2. Navigate to Insights and then to Community Standards

3. Select Add on the license row

No license no reuse

1. Go to any of your own public GitHub repos with no license
2. Navigate to Insights and then to Community Standards
3. Select Add on the license row

E1. Adding a missing license to your public repo

4. You will see the license templates

Add a license to your project

Apache License 2.0
GNU General Public License v3.0
MIT License
BSD 2-Clause "Simplified" License

Choose a license to add to your project

Select a template on the left to get started.

Learn more about [which license best fits your project](#).

5. Choose the one you want. We use MIT in the example.

- Remember that <https://choosealicense.com> could help you choose
- Review and submit

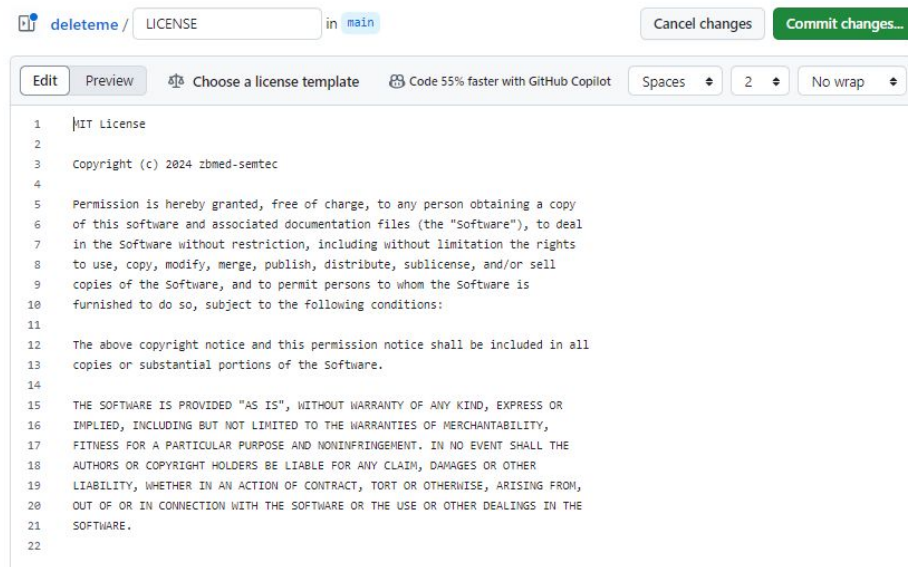
Add a license to your project

Apache License 2.0	<p>A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.</p> <p>Permissions</p> <ul style="list-style-type: none">✓ Commercial use✓ Modification✓ Distribution✓ Private use <p>Limitations</p> <ul style="list-style-type: none">✗ Liability✗ Warranty <p>Conditions</p> <ul style="list-style-type: none">① License and copyright notice <p>This is not legal advice. Learn more about repository licenses.</p> <p>MIT License</p>	<p>To adopt MIT License, enter your details. You'll have a chance to review before committing a LICENSE file to a new branch or the root of your project.</p> <p>Year ①</p> <input type="text" value="2024"/> <p>Full name ①</p> <input type="text" value="zbmed-semantic"/> <p>Review and submit</p>
GNU General Public License v3.0		
MIT License		
BSD 2-Clause "Simplified" License		
BSD 3-Clause "New" or "Revised" License		

Boost Software License 1.0

E1. Adding a missing license to your public repo

6. Commit changes and now you have a license!



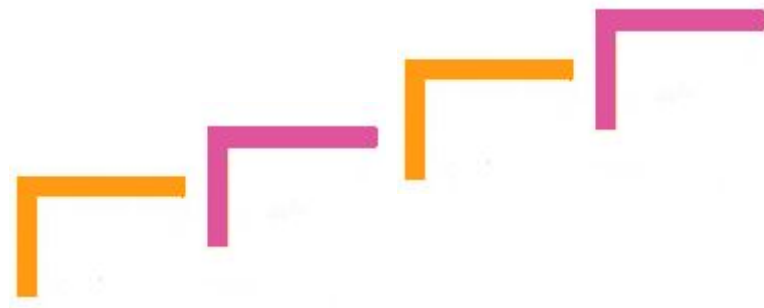
The screenshot shows a code editor interface for a file named `LICENSE` in the `main` branch. The editor has a top bar with buttons for `Edit`, `Preview`, `Choose a license template`, and `Code 55% faster with GitHub Copilot`. The main area displays the MIT license text, which is being edited. The text is as follows:

```
1 |MIT License
2
3 Copyright (c) 2024 zbmed-semtec
4
5 Permission is hereby granted, free of charge, to any person obtaining a copy
6 of this software and associated documentation files (the "Software"), to deal
7 in the Software without restriction, including without limitation the rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
22
```


E2. Adding a missing license to your (private or public) repo

No license no reuse

1. Go to any of your own GitHub repos with no license
2. On the landing page, choose to create a new file
3. Use the file name LICENSE
4. You will see the license templates → GitHub will automatically give you the option to use a license template
5. Choose the one you want
6. Commit changes and now you have a license!

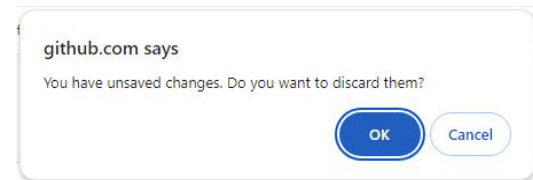
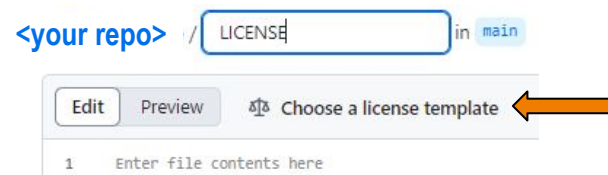
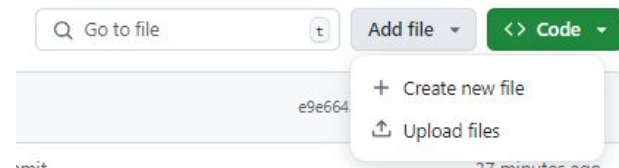


Step by step in the following slides

E2. Adding a missing license to your (private or public) repo

No license no reuse

1. Go to any of your own GitHub repos with no license
2. On the landing page, choose to create a new file
3. Use the file name LICENSE
 - GitHub will automatically give you the option to use a license template
 - Warning: as you are “abandoning” changes on the new file, it will show a pop-up. Click OK so you go to the license templates



E2. Adding a missing license to your (private or public) repo

4. You will see the license templates

Add a license to your project

Apache License 2.0
GNU General Public License v3.0
MIT License
BSD 2-Clause "Simplified" License

Choose a license to add to your project

Select a template on the left to get started.

Learn more about [which license best fits your project](#).

5. Choose the one you want, for instances, MIT as shown in the example.

- Remember that <https://choosealicense.com> could help you choose
- Review and submit

Add a license to your project

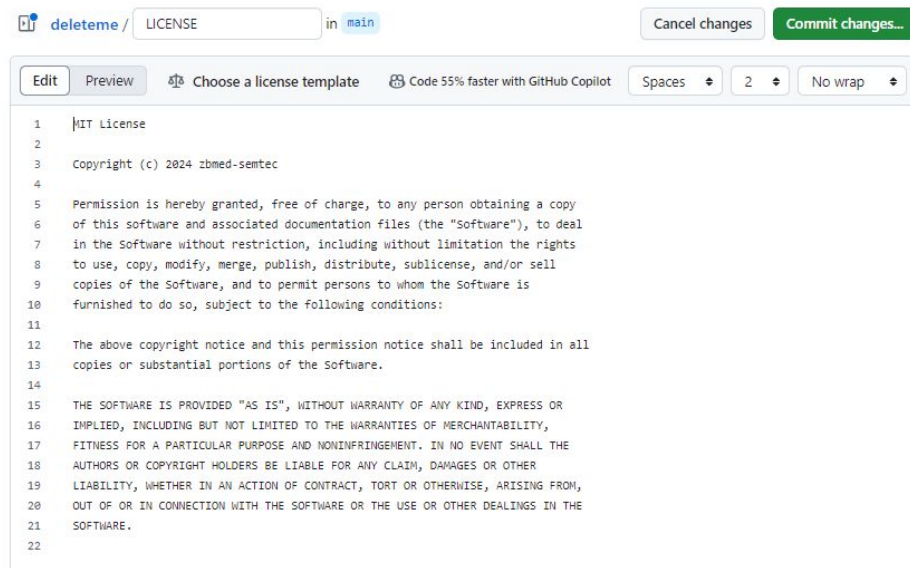
Apache License 2.0	<p>A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.</p> <p>Permissions</p> <ul style="list-style-type: none">✓ Commercial use✓ Modification✓ Distribution✓ Private use <p>Limitations</p> <ul style="list-style-type: none">✗ Liability✗ Warranty <p>Conditions</p> <ul style="list-style-type: none">① License and copyright notice <p>This is not legal advice. Learn more about repository licenses.</p>	<p>To adopt MIT License, enter your details. You'll have a chance to review before committing a LICENSE file to a new branch or the root of your project.</p> <p>Year ①</p> <input type="text" value="2024"/> <p>Full name ①</p> <input type="text" value="zbmed-semantic"/> <p>Review and submit</p>
GNU General Public License v3.0		
MIT License		
BSD 2-Clause "Simplified" License		
BSD 3-Clause "New" or "Revised" License		

Boost Software License 1.0

MIT License

E2. Adding a missing license to your (private or public) repo

6. Commit changes and now you have a license!



The screenshot shows a code editor interface for adding a license to a repository. At the top, there's a header bar with a file icon, the text "delete.me / LICENSE", and "in main". To the right are two buttons: "Cancel changes" and "Commit changes...". Below the header bar, there's a toolbar with "Edit", "Preview", "Choose a license template", and "Code 55% faster with GitHub Copilot". The main area shows a license template with line numbers 1 through 22. The text of the license is as follows:

```
1 |MIT License
2
3 Copyright (c) 2024 zbmed-semtec
4
5 Permission is hereby granted, free of charge, to any person obtaining a copy
6 of this software and associated documentation files (the "Software"), to deal
7 in the Software without restriction, including without limitation the rights
8 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9 copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
22
```

E3. Create or update your CFF

Generate your citation metadata files with ease

CITATION.cff files are plain text files with human- and machine-readable citation information for software and datasets.

Code developers can include such files in their source code repositories to let others know how to correctly cite their software.

You can read more about the Citation File Format in the [official CFF specification website](#).



Create your CITATION.cff now, or start from an existing file!




+ Create

Update

Version 2.3.1

<https://citation-file-format.github.io/cff-initializer-javascript>

Let's improve our CFFs → website

- Go to CFFInit
- Use  Update
- Go to any of your code repositories and copy the content of the CFF file
- Paste it and  PARSE
- Have a look to the results, edit it as needed to get a valid (or improved) CFF
- Do not have a CFF at hand? Give it a try!  Create

⚠ cff-version was updated to 1.2.0. This might led to some issues, so verify before downloading.

✓ Parsed CFF successfully.

E3. Is your CFF correct?

Validation

You can validate your `CITATION.cff` file on the command line with the [cffconvert](#) Python package:

```
# Install cffconvert with pip in user space
python3 -m pip install --user cffconvert

# Validate your CFF file
cffconvert --validate
```

If you get a Traceback with error messages, look for the relevant validation error and fix it. If the output is very long, it may help if you search it for lines starting with `jsonschema.exceptions.ValidationError`.

If you prefer to use Docker, you can use the [cffconvert](#) Docker image:

```
cd <directory-containing-your-CITATION.cff>
docker run --rm -v ${PWD}:/app citationcff/cffconvert --validate
```

https://github.com/citation-file-format/citation-file-format/blob/main/README.md#validation-heavy_check_mark

```
validation.invalid
--- All found errors ---
["Cannot find required key 'date-released'. Path: ''"]
pykwalify.errors.SchemaError: <SchemaError: error code 2: Schema validation failed:
- Cannot find required key 'date-released'. Path: '': Path: '/'>
PS C:\workspace\zbmed\BioHackOutcomes>
```

Let's improve our CFFs

- Choose a **local** GitHub repo you will use for this exercise
- Go to the local folder containing the CFF folder
- Run the Docker version of the CFF validator
 - `docker run --rm -v ${PWD}:/app citationcff/cffconvert --validate`
- You can also try the pip option
 - `python3 -m pip install --user cffconvert`
 - `cffconvert --validate`
- How did it work for you?

E4. Metadata extraction for research software

Software Metadata Extraction Framework

(SOMEF)     

Authors: Daniel Garijo, Allen Mao, Miguel Ángel García Delgado, Haripriya Dharmala, Vedant Diwanji, Jiaying Wang, Aidan Kelley and Jenifer Tabita Ciuciu-Kiss.

The aim of SOMEF is to help automatically extract metadata from scientific software from their readme files and GitHub repositories and make it available in a machine-readable manner.

Thanks to SOMEF, we can populate knowledge graphs of scientific software metadata and relate different software together.

SOMEF has currently been tested with GitHub repositories, but it can extract metadata from any readme file written in markdown syntax.

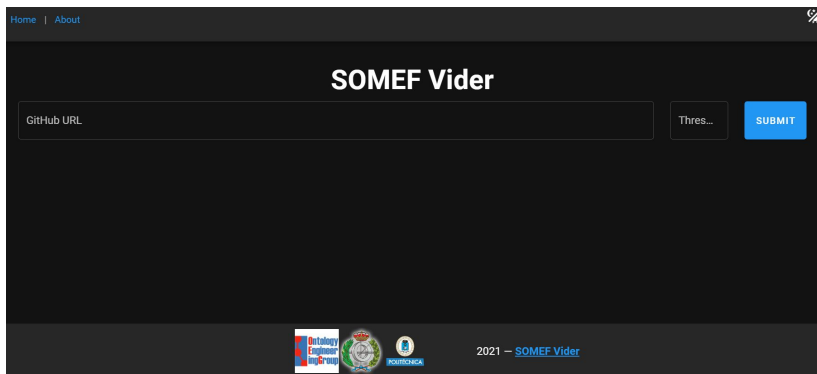
Info

If you experience any issues when using SOMEF, please open an issue on our [GitHub repository](https://github.com/zbmed-semtec/somef).

<https://somef.readthedocs.io/>

- Choose a GitHub repo you will use for this exercise
- (optional) Create a venv
- Install SOMEF and (optional for SOMEF format) configure prerequisites
 - `pip install somef`
 - `python -m nltk.downloader wordnet`
 - `python -m nltk.downloader omw-1.4`
 - `somef configure -a`
- Do not install, rather use docker
 - `docker run --rm -v ${PWD}:/app -it kcapd/somef /bin/bash`
- Test it is working
 - `somef --help`
- Generate a JSON file (SOMEF format)
 - `somef describe -r https://github.com/zbmed-semtec/BioHackOutcomes/ -o test.json -t 0.8 -p`
- Generate a JSON file (CodeMeta format)
 - `somef describe -r https://github.com/zbmed-semtec/BioHackOutcomes/ -c test_codemeta.json -p`

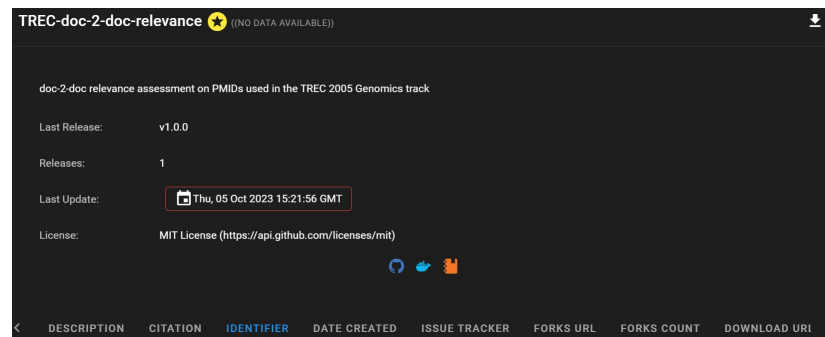
E4. Metadata extraction for research software

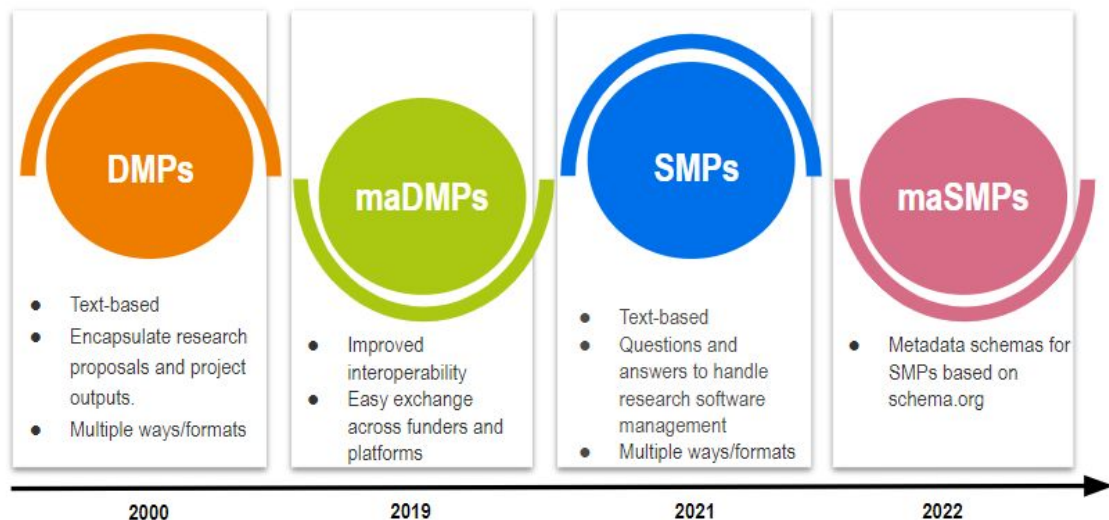


<https://github.com/SoftwareUnderstanding/SOMEF-Vider>, deployed at <https://somef.linkeddata.es/>

Let's get some metadata

- Visit <https://somef.linkeddata.es/>
- Try it out with any public GitHub repo
- What metadata do you get?

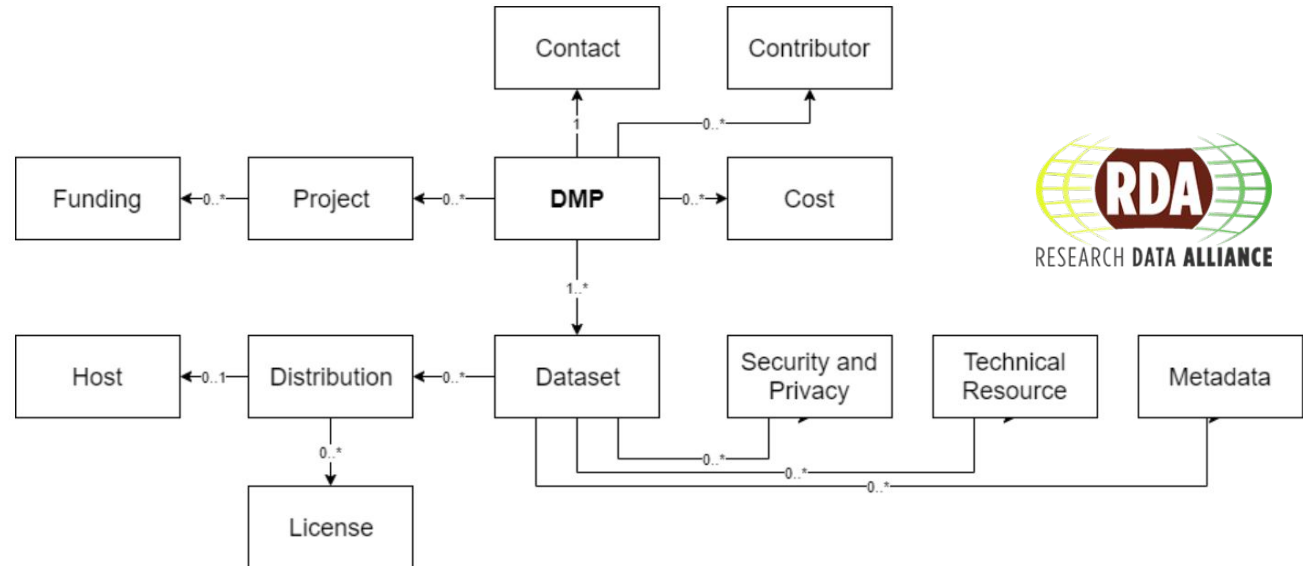






Software Management Plan

"Add semantics to the SMP requirements as a way to improve accessibility and usability of research software in life sciences."



Data Management Plans • RDMO for MPG

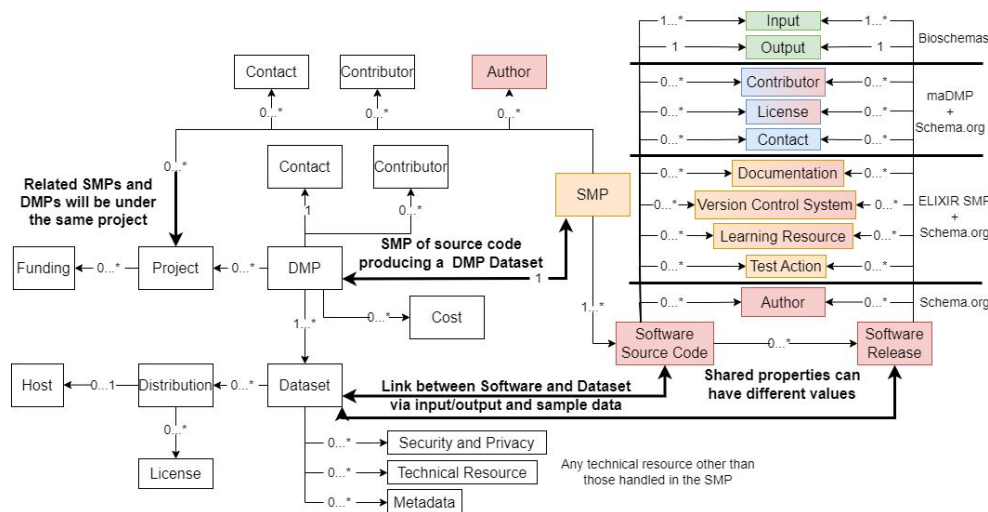
Software Management Plan Template for the RDMO now available

Core requirement (Section 5.1)	Example SMP question(s) (Section 6.1)
Purpose	Please provide a brief description of your software, stating its purpose and intended audience.
Version control	How will you manage versioning of your software?
Repository	How will you make your software publicly available? If you do not plan to make it publicly available you should provide a justification.
User documentation	How will your software be documented for users? Please provide a link to the documentation if available. How will you document your software's contribution guidelines and governance structure?
Software licencing and compatibility	What licence will you give your software? How will you check that it respects the licences of libraries and dependencies it uses?
Deployment documentation	How will the installation requirements of your software be documented? Please provide a link to the installation documentation if available.
Citation	How will users of your software be able to cite your software? Please provide a link to your software citation file (CFF) if available.
Developer documentation	How will your software be documented for future developers?
Testing	How will your software be tested? Please provide a link to the (automated) testing results.
Software Engineering quality	Do you follow specific software quality guidelines? If yes, which ones?
Packaging	How will your software be packaged and distributed? Please provide a link to available packaging information (e.g. entry in a packaging registry, if available).
Maintenance	How do you plan to procure long term maintenance of your software?

<div> <div> <div></div> <div>Software Management Plan</div> </div> <div> <div> <div></div> <div>Accessibility & License</div> </div> <div> <div>What is the name of the software?</div> <div>How can the software be accessed by third parties?</div> <div>Does your software have a license?</div> </div> <div> <div> <div></div> <div>Documentation</div> </div> <div> <div>What type of documentation is available, provided with the software?</div> <div>Is the purpose of the software stated in the documentation?</div> <div>Does the documentation describe how to use the software?</div> </div> <div> <div> <div></div> <div>Testing</div> </div> <div> <div>What type of testing do you use?</div> <div>Are sample data and/or parameters that can be used to test the software?</div> </div> <div> <div> <div></div> <div>Interoperability</div> </div> <div> <div>Do you use well-established standard input/output formats?</div> <div>What programming languages are you using in your project?</div> </div> <div> <div> <div></div> <div>Versioning</div> </div> <div> <div>Do you use a version control system?</div> <div>Do you use Semantic Versioning?</div> </div> </div> </div> </div> </div></div></div>

Source: <https://doi.org/10.5281/zenodo.7248877>Source: <https://smw.ds-wizard.org/>

Repo and web pages, enrichment cycle [10.5281/zenodo.10374838](https://zenodo.org/record/10374838) with SOMEF, RDMO and maSMP (types and properties and usage guidance, aka profiles)



machine-actionable Software Management Plans

DOI [10.5281/zenodo.7806638](https://doi.org/10.5281/zenodo.7806638) DOI [10.5281/zenodo.10582121](https://doi.org/10.5281/zenodo.10582121)

Version **2.1.0** Schema **maSMP** Release **2.1.0**

Profiles **2.1.1** Schema **maSMPProfile** Release **2.1.1**

<https://zbmed-semtec.github.io/maSMPs>

<https://github.com/zbmed-semtec/maSMPs>

Metadata elements for a maSMP

Software Source Code (aka SoftwareSourceCode in schema.org)		Software Release (aka SoftwareApplication in schema.org)	
Property name	Possible values (range)	Property name	Possible values (range)
identifier	PropertyValue, Text, URL	identifier	PropertyValue, Text, URL
name	Text	name	Text
description	Text	description	Text
license	Text, URL	license	Text, URL
author	Organization or Person	author	Organization or Person
contributor	Organization or Person	contributor	Organization or Person
citation	CreativeWork, Text, URL	citation	CreativeWork, Text, URL
conditionsOfAccess	Text	conditionsOfAccess	Text
isAccessibleForFree	Boolean	isAccessibleForFree	Boolean
codeRepository	URL	releaseNotes	Text, URL
programmingLanguage	ComputerLanguage, Text	memoryRequirements	Text
targetProduct (aka Software Release)	SoftwareApplication	operatingSystem	Text
archivedAt	URL	processorRequirements	Text
discussionURL	URL	storageRequirements	Text
usageInfo	CreativeWork, URL	supportingData	Dataset
version (i.e., semantic version)	Text	version (i.e., semantic version)	Text
hasContact	Organization or Person	hasContact	Organization or Person
input	FormalParameter, Dataset	input	FormalParameter, Dataset
output	FormalParameter, Dataset	output	FormalParameter, Dataset
hasAPIDocumentation	Documentation	hasAPIDocumentation	Documentation
hasDeveloperDocumentation	Documentation	hasDeveloperDocumentation	Documentation
hasUserDocumentation	Documentation	hasUserDocumentation	Documentation
hasLearningResource	LearningResource	hasLearningResource	LearningResource
hasVersionControlSystem	SoftwareApplication	hasVersionControlSystem	SoftwareApplication
hasReadme	URL	hasReadme	URL
testedWith	TestAction	testedWith	TestAction





From schema.org

From maDMP

From Bioschemas

From maSMP
(New elements)

maSMPs types and properties

Definitions Change View ↕			
Date Added	Last Updated	Schema Status	Schema Source
December 4th 2023	January 29th 2024	schema ok	View Source
OutputManagementPlan > 			
Management plans (e.g., data, software, other research output/outcomes/artifacts) are a tool for researchers and research stewards to organize and plan around research artifacts, from initial ideas to publication. This type builds on top of DataCite Metadata Schema 4.4 and maps it mandatory elements to schema.org to represent research artifact management plans. This type could be extended to cover particularities wrt management plans for specific research outcomes, e.g., data management plans or software management plans.			
SoftwareRunAction > 			
The act of testing an object according to its specifications. For instance, testing a software with a particular testType using a specific testInput and getting a specific testOutput (aka result).			
SoftwareTestAction > 			
The act of testing an object according to its specifications. For instance, testing a software with a particular testType using a specific testInput and getting a specific testOutput (aka result).			
SoftwareManagementPlan > 			
Software Management plan.			

maSMPs profiles

Definitions Change View ↕			
Date Added	Last Updated	Schema Status	Schema Source
January 29th 2024	January 29th 2024	schema ok	View Source
<div><div>SoftwareRunActionProfile ></div><div>The act of testing an object according to its specifications. For instance, testing a software with a particular testType using a specific testInput and getting a specific testOutput (aka result).</div></div>			
<div><div>SoftwareTestActionProfile ></div><div>The act of testing a software according to its specifications. For instance, testing a software with a particular testType using a specific testInput and getting a specific testOutput (aka result).</div></div>			
<div><div>SoftwareManagementPlanProfile ></div><div>Software Management plan.</div></div>			
<div><div>SoftwareApplicationProfile ></div><div>Profile of schema:SoftwareApplication using properties corresponding to the Software Management Plan use case.</div></div>			
<div><div>SoftwareSourceCodeProfile ></div><div>Profile of schema:SoftwareSourceCode using properties corresponding to the Software Management Plan use case</div></div>			

Metadata related to ML

- Training Datasets
- Software to train and optimize the model
- **ML models**
- Extrinsic cross-validation/evaluation
- Deployment → easy to use



MLentory



maPlan

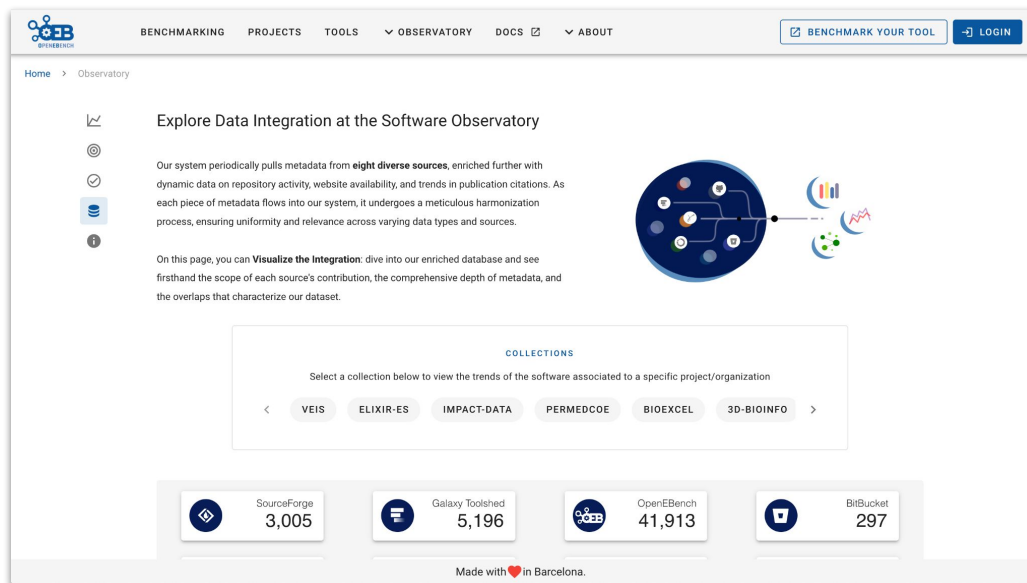
Metadata related to DMPs

- ZB MED/NFDI4DS maSMP, RDA maDMP and DataCite OutputManagementPlan as starting points
- maDMP and maSMP Integration to RDMO
- Deployment → easy to use

Goal: Let's build together the core metadata for a good FAIR data for AI management plan that enable/facilitate the creation of Knowledge Graphs for research software

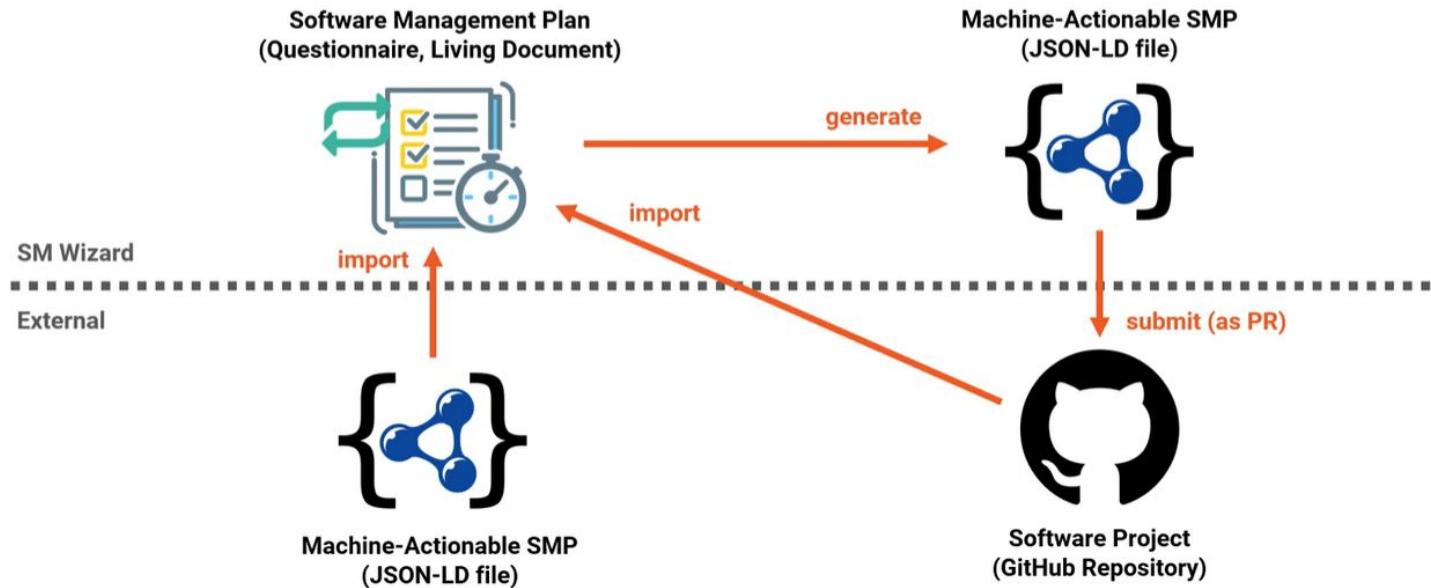


Tool to systematically monitor and assess the quality of Life Sciences Research software



<https://openebench.bsc.es/observatory>

Workflow with SM Wizard



This tutorial includes information published at

- Alves R, Bampalikis D, Castro LJ, González JMF, Harrow J, Kuzak M, et al. ELIXIR Software Management Plan for Life Sciences. BioHackrXiv; 2021. doi:10.37044/osf.io/k8znb
- Hong NC, Katz DS, Barker M, Lamprecht A-L, Martinez C, Psomopoulos FE, et al. FAIR Principles for Research Software (FAIR4RS Principles) | RDA. 2021. doi:10.15497/RDA00065
- Barker M, Chue Hong NP, Katz DS, Lamprecht A-L, Martinez-Ortiz C, Psomopoulos F, et al. Introducing the FAIR Principles for research software. Sci Data. 2022;9: 622. doi:10.1038/s41597-022-01710-x
- Castro Leyla Jael, Giraldo Olga, Geist Lukas, Quiñones Nelson, Solanki Dhvani, & Rebholz-Schuhmann Dietrich. (2024). machine-actionable Software Management Plan Ontology (maSMP Ontology) (2.1.0). Zenodo. <https://doi.org/10.5281/zenodo.10582073>
- Castro, L. J., Giraldo, O., Geist, L., Quiñones, N., Solanki, D., & Rebholz-Schuhmann, D. (2024). Usage guidance (aka profiles) for the machine-actionable Software Management Plan Ontology (2.1.1). Zenodo. <https://doi.org/10.5281/zenodo.10582121>
- Aidan Kelley, Daniel Garijo; A framework for creating knowledge graphs of scientific software metadata. Quantitative Science Studies 2022; 2 (4): 1423–1446. doi: https://doi.org/10.1162/qss_a_00167

This tutorial includes information from other tutorials

- Castro, L. J. (2024, February 7). Adding Bioschemas Dataset and ComputationalTool markup to GitHub pages. Zenodo. <https://doi.org/10.5281/zenodo.10629453>
- Castro, L. J., Soiland-Reyes S., Grieb J., Weiland C. (2024, March 19). Practical web-based FDOs with RO-Crate and FAIR Signposting. Presented at FDO Summit 2024. Available at <https://drive.google.com/file/d/1TKxFpz4QcG9bGCrGOB8Zq9kZU-H0kc6S/view>

Images

- ZB MED, taken from internal documents
- NFDI4DataScience, taken from <https://www.nfdi4datascience.de/contact/>
- AIKG-SD, taken from <https://sites.google.com/view/aikg-sd-summer-school-day-2024/startseite>
- EOSC Future, taken from <https://eoscfuture.eu/wp-content/uploads/2021/09/EOSC-Future-Brand-Guidelines.pdf>
- European Union flag, taken from https://european-union.europa.eu/principles-countries-history/symbols/european-flag_en
- DFG, modified from <https://www.dfg.de/de/service/logo-corporate-design>
- KI 2024, taken from <https://www.informatik.uni-wuerzburg.de/ki24/>
- FORCE11, taken from <https://force11.org/>
- RDA, taken from <https://www.rd-alliance.org/>
- ReSA, taken from <https://www.researchsoft.org/>
- Semantic Web, taken from <https://www.w3.org/2001/sw/>
- JSON-LD, taken from <https://json-ld.org/images/>
- RDF, taken from https://commons.wikimedia.org/wiki/File:Rdf_logo.svg
- PubMed screenshot, taken from <https://pubmed.ncbi.nlm.nih.gov/>
- OpenAlex screenshot, taken from <https://openalex.org/>
- Semantic spectrum, taken from <https://doi.org/10.1145/956863.956932>
- GitHub metadata, taken from <https://github.com/zbmед-semtec/BioHackOutcomes>
- Zenodo metadata, taken from <https://zenodo.org/records/7341391>
- SOMEF screenshot, taken from <https://somef.readthedocs.io/en/stable/>
- SOMEF VIDER screenshot, taken from <https://somef.linkeddata.es/>
- GitHub, SVG taken from <https://github.com/>
- GitLab, SVG taken from <https://gitlab.com>
- Zenodo, taken from <https://zenodo.org/>
- SWH archive, taken from <https://archive.softwareheritage.org/>
- ORCID, taken from <https://orcid.org/>
- CFF, partial screenshot taken from <https://citation-file-format.github.io/> and <https://citation-file-format.github.io/cff-initializer-javascript>
- bio.tools, partial screenshot taken from <https://bio.tools/>
- NFDI, taken from <https://www.nfdi.de/downloads>
- Research Software Directory, partial screenshot taken from <https://research-software-directory.org/>
- Schema.org, partial screenshot from <https://schema.org>
- Bioschemas, partial screenshot from <https://bioschemas.org>
- schemas.science, modification of Bioschemas and partial screenshot from <https://schemas.science/>
- ELIXIR, modified from <https://elixir-europe.org/services/compute/aai/login-guidelines>
- maDMP diagram, taken from <https://github.com/RDA-DMP-Common/RDA-DMP-Common-Standard>
- eScience Centre SMP, taken from <https://doi.org/10.5281/zenodo.7248877>
- ELIXIR SMP, partial screenshot taken from <https://smw.ds-wizard.org/>
- MPDL SMP, partial screenshot take from <https://rdm.mpg.de/2022/12/09/smp-template-available/>
- ZB MED maSMP types, partial screenshot taken from <https://discovery.biothings.io/ns/maSMP>
- ZB MED maSMP profiles, partial screenshot taken from <https://discovery.biothings.io/ns/maSMPPProfiles>
- maSMP, MLentory and maPlan logos CC-BY 4.0, credits to Leyla Jael Castro
- We use free SVG icons from [Font Awesome](#)

This work has been partially funded by the German Research Foundation (DFG) through the grant for NFDI4DataScience No. 460234259

Thanks! Danke!

If you use this tutorial/training material, please cite it as

Castro, L. J. (2024, September 19). Introduction and tutorial on Metadata and SMPs for FAIR research software. 47th German Conference on Artificial Intelligence (KI 2024), Würzburg, Germany. Zenodo. <https://doi.org/10.5281/zenodo.13799879>

Special thanks to the SemTec team at ZB MED, in particular authors and contributors behind the maSMP metadata schema (see <https://zbmed-semtec.github.io/maSMPs/about/>)

Leyla Jael Castro
Semantic Retrieval Team at [ZB MED](#)
[NFDI4DataScience](#)