FAIRmaterials: Ontology Tools with Data FAIRification in Development

26 June 2024

Summary

The bilingual `FAIRmaterials` package is a robust and user-friendly tool designed to simplify the creation and visualization of materials and data science ontologies. `FAIRmaterials`, available in the Python and R languages, addresses the complexities and technical barriers associated with traditional ontology editors based on manual user input such as `Protege` (Musen 2015). `FAIRmaterials` offers an intuitive workflow and easy-to-use templates, making it accessible to users both experienced and inexperienced with ontology creation. By leveraging a CSV-based template system, the package allows for the straightforward design and expansion of ontologies, promoting better data accessibility, management, and interoperability in research environments.

The core feature of the `FAIRmaterials` package is its ability to programatically convert simple and structured CSV inputs into rich, well-defined ontologies. This capability is designed to support the findability, accessibility, interoperability, and reusability (FAIR) (Wilkinson et al. 2016) of research data and serve as a tool in the process of data FAIRification. `FAIRmaterials` also integrates seamlessly with other existing data systems and protocols within the scientific community. The package supports various serialization formats, such as Turtle and JSON-LD, to accommodate different user preferences and technical needs.

Designed with the end-user in mind, `FAIRmaterials` facilitates a comprehensive yet entry-level experience in ontology creation. Its additional features, such as automated ontology merging, static visualizations, and comprehensive documentation for outputs extend its utility, making it a valuable tool for any researcher engaged in knowledge management.

Statement of need

`Protege` is currently the most widely-used open-source tool for ontology creation and development. Its main capabilities include manually creating and editing ontological terms and relationships, visualizing ontologies, checking the logical consistency of ontologies, and querying ontologies for specific information. While Protege's extensive functionality even facilitates plugins, the complexity of the interface is a barrier for those who have little experience with ontology creation. This complexity prevents certain researchers from creating and integrating ontologies with their own datasets entirely. Therefore, there is a need for a tool that can create ontologies with an interface that is easily understandable and provides ample documentation on how to use it. `FAIRmaterials` seeks to lower the barrier of entry for scientists entering the world of ontology development and evolution. The package provides a baseline CSV ontology template with built-in and easy-to-follow instructions on how to design an ontology which can be found here.

The CSV format was chosen as an input for `FAIRmaterials` due to its well-established familiarity within the scientific community as well as its ease of use without the need for specialized tools or programs. The primary function of our package is to programmatically convert the filled-out CSV files into a well-structured ontology file. In addition to outputting the ontology in multiple syntaxes, the package outputs other files to aid in the

ontology development and publishing process, such as a static visualization and an HTML documentation file about the ontology.

Variable Name	Belongs to Ontology	Parent Variable	Definition of Variable	Alternative Name(s)	Unit	Logical Axioms
The variable that you would like to represent in your ontology schema.	If the variable exists in an ontology below, select the ontology from the dropdown menu. Otherwise, please leave blank.	Please type the variable from the Parent Ontology that you would like to connect your variable to. If the variable is already in a selected ontology, please leave blank.	Please provide the definition the variable. For reccomendation on a definition, search the term in https://schema.org. If the variable is already in a selected ontology, please leave blank.	Please provide any alternative names for the variable in the literature. If the variable is already in a selected ontology, please leave blank *Optional*	Please provide the unit that the variable is expressed in your data. For a dictionary of standardized units, please visit https://qudt.org. If the variable is already in a selected ontology, please leave blank *Optional*	Please provide any logic you would like to attach to your variable. For information and examples on logical axioms, please visit https://www.w3.org/TR/daml+oil-axioms/ If the variable is already in a selected ontology, please leave blank. *Optional*

Figure 1: Empty Variable CSV Template Sheet for the `FAIRmaterials` Package. The CSV template sheet includes specific instructions on how to fill out every row to correctly generate the ontology. Template is split in half for readability.

Key Features

Ontology creation from template CSVs

The primary function of the FAIRMaterials package is to convert the term, relationship, and value specifications from the CSV template into an ontology. Both the Python and R versions of the package rely on an open-source library called `RDFLib` (Carl Boettiger, n.d.; RDFLib Development Team, n.d.), which allows users to programmatically create and populate an ontology, serialize it into many syntaxes, validate its logical consistency, and provide output documentation. The package parses through the CSV sheets and generates an `RDFLib` graph object of the ontology. Both versions of the package also have an add_external_onto_info argument or flag that facilitates the updating of information from the CSV sheets by parsing each external term‘s ontology. Importing this information simplifies the process of filling out the sheets, as users do not need to search for this information themselves. An overview of the sheets header is illustrated in Figure 1.

Ontology serialization into multiple syntaxes

Once an `RDFLib` graph object is created, the package automatically generates an output folder and serializes the object into two syntaxes: Turtle and JSON-LD. The ontology is serialized into two syntaxes because of the unique advantages that each syntax provides. One of the main advantages of Turtle is that it is easy-to-read by both humans and machines without the need for specialized translators, and it has the syntax requirements to fully represent the complex relationships that Turtle file facilitates. The distinct advantage of JSON-LD is that it is a very common form of linked data in the scientific community, a familiarity that makes it a prime choice for sharing with other researchers and allowing them to make modifications to the ontology, such as adding additional alternative labels to concepts.

Static visualization output of ontology

Determining the correctness of an ontology is difficult if its representation is in a textual format. For this reason, the package outputs a visualization in both the R and Python versions. The optional Python flag include_graph_valuetype can be used to include value type nodes in the output visualization. In R, include_graph_valuetype is an argument and can be set to true or false, removing the value type nodes can be helpful to when trying to visualize large ontologies. The Python version uses the `Graphviz` (Graphviz Development Team, n.d.) software to create a graph object at the same time as the `RDFLib` graph object. The package then outputs the Graphviz object as a PNG (as illustrated in Figure 2 so that users can immediately visualize the ontology and make corrections to the sheet if there are inconsistencies. The color schema and overall design of the `Graphviz`; PNG is modeled after the popular `WebVOWL` (Lohmann et al. 2015) ontology visualization tool to make it easier for users to inherently understand the color schema and format. The R version generates a visualization using the `DiagrammeR` (Iannone and Roy, n.d.) package available on the Comprehensive R Archive Network (CRAN), as `Graphviz` is not available in the R language. The output is saved as an SVG for easy visualization with a similar color scheme to the Python package and `WebVOWL` tool.

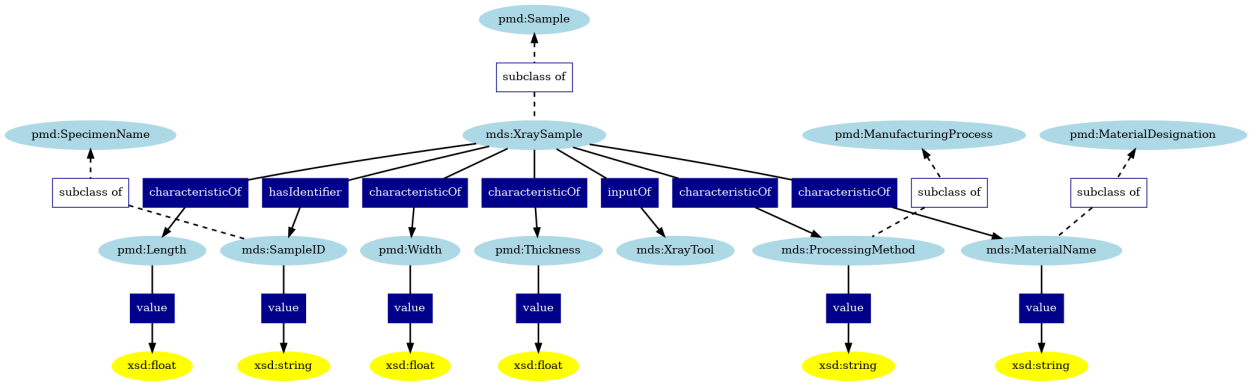

Figure 2: The X-ray sample ontology. The light-blue icons represent ontology terms, with the prefix (i.e. pmd) indicating the ontology that the term was created in. The dark-blue squared boxes indicate relationships created between entities. The yellow round boxes either indicate the type of the value stored in each subclass or the unit that the value is expressed in, with the prefix indicating the ontology the unit definition belongs to or the schema language that the value type is defined in.

Ontology merging

Both the R and Python versions of the `FAIRmaterials` package feature an ontology merging capability. The package processes all CSV files within a specified folder and its subdirectories, which will be merged into one ontology. For each subdirectory containing a complete set of CSV sheets, the package generates separate, unmerged outputs. A merged output incorporating all ontologies is created in the main folder path. The merged output can include customized metadata such as title, authors, version, URI, and description. These are included by using the merge_base_uri, merge_title, merge_author, merge_version, and merge_description arguments in R or as a flag in Python. The merge_base_uri must be specified for the merged ontology; other metadata elements may be left to a default. The R version defaults to concatenating the metadata from the included ontologies and the Python version has preset default values.

Corresponding documentation output for ontology

One important aspect of ontologies is that they are easily readable by humans as well as machines. Not only do the Turtle and JSON-LD formats have this dual readability, but the HTML documentation also provides a more intuitive interface for humans to understand the terms and relationships stored in ontologies. The Python version of the package leverages this by using `RDFLib` to output a PyLode HTML file, which consists of user-friendly documentation about the output ontology file. Unfortunately, the R version does not have the same capability because the R version of the RDFlib package does not create HTML files.

Typical Usage

In order to use the `FAIRmaterials` package effectively, a user must conceptualize their ontology, fill out the CSV template, run the package, and finally review the output ontology.

Design the ontology schema

It is recommended that users first design an ontology schema that includes all the vocabulary needed to describe a particular dataset. Creating an ontology schema helps to ensure that the ontology has explicit and well-thought-out connections to the Basic Formal Ontology (BFO) or another top level ontology ensuring its interoperability with other, existing ontologies. It is recommended that every variable in the ontology schema is tagged as a subclass of an already-existing ontology term. Other top level terms should be used within the schema when necessary, such as using a QUDT ontology term when associating a certain measurement term with a standardized unit. An example of an ontology schema is showed in Figure 3.

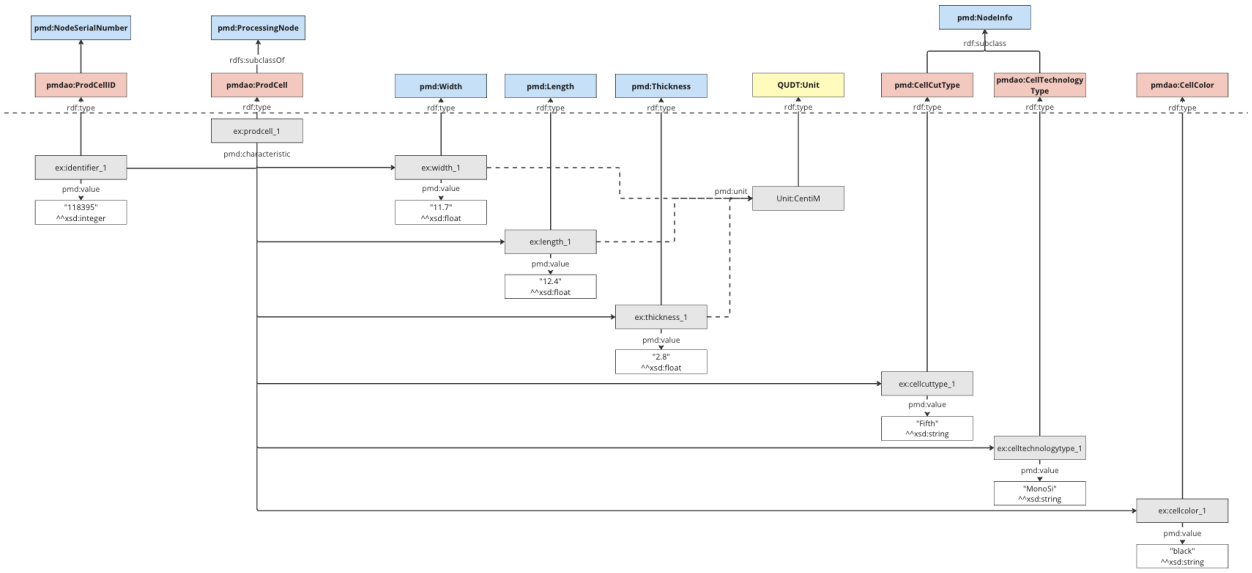

Figure 3: Example of Schema.

Complete the CSV template

Users must complete the provided CSV template to input the details of their ontology. The CSV template includes instructions on how to do this, and notifies the user which fields are optional.

How to run the package

In Python

To run the `FAIRmaterials` package in Python, users should execute it from the command line using the following syntax: FAIRmaterials –options. Users must specify the directory containing the CSV sheets using the –folder_path flag and a –merge_base_uri flag if there are two or more sets of ontology files. To process the files from a generic path:

```
python -m FAIRmaterials --folder_path /path/to/csv/files  --include_graph_valuetype
```

There are several flags such as –include_graph_valuetype, –include_pylode_docs, –add_external_onto_info, and flags for adding merged ontology meta data. To process the files with flags ensuring the package generates a pylode adds external information and specifies some ontology metadata:

```
python -m FAIRmaterials --folder_path /path/to/csv/files/
    --include_graph_valuetype --include_pylode_docs
    --add_external_onto_info --merge_title ExampleOnto
    --merge_base_uri https://example.org/
    --merge_version 1.0
```

The package will then create an ontology graph, saved in PNG format, serialize the RDF object to both Turtle and JSON-LD formats, and create a pylode HTML.

In R

To execute the package in R use the process_ontology_files function. Users must specify the directory containing the CSV sheets using the folder_path argument and the merge_base_uri argument if there are two or more sets of ontology files. To process the files from a generic path:

```r
FAIRmaterials::process_ontology_files("/path/to/csv/files")
```

There are several arguments such as include_graph_valuetype, add_external_onto_info, and more for adding merged ontology meta data. To process the files with arguments ensuring the package adds external information and specifies some ontology metadata:

```r
FAIRmaterials::process_ontology_files("/path/to/csv/files",
    include_graph_valuetype = TRUE,
    add_external_onto_info = TRUE,
    merge_title = "ExampleOnto",
    merge_base_uri = "https://example.org/",
    merge_version = "1.0")
```

The package processes these files to create an ontology graph, saved in SVG format, and serializes the RDF object to both Turtle and JSON-LD formats.

Finalize and review the output

It is recommended that users carefully examine the generated ontology files and visualizations and make any necessary adjustments to the CSV template based on this review to correct potential errors or to refine the ontology structure. Finally, after review, the user is left with JSON-LD, Turtle, and visualization files that are not only machine-readable, but also human-friendly, and therefore enhance the shareability and usability of the user's research. With this streamlined workflow, researchers at all levels of experience can now utilize ontology development with the `FAIRmaterials` package. The package allows researchers to focus less on the difficulties of ontology creation and more on their scientific investigations by simplifying the technical aspects of ontology creation.

Code Availability

The `FAIRmaterials` Python package can be easily accessed from The Python Package Index (PyPI) (Python Software Foundation, n.d.). To install the package, simply search for `FAIRmaterials` on the PyPI website or click here and follow the provided instructions for installation.

The package can also be installed with the following Python command:

```
pip install FAIRmaterials
```

The `FAIRmaterials` R package can be easily accessed on the Comprehensive R Archive Network (CRAN) (R Project, n.d.). To install the package, simply search for `FAIRmaterials` on the CRAN website or click here and follow the provided instructions for installation. The package can also be installed with the following R command:

```r
install.packages("FAIRmaterials")
library(FAIRmaterials)
```

The package includes comprehensive documentation, including detailed usage examples. These can be found in the packages vignette available on the `FAIRmaterials` page on CRAN found here. You can also access the package vignette after installation using the following R command:

```r
vignette("FAIRmaterials")
```

More documentation can be found after installation using the following R command:

```r
?process_ontology_files
```

The code for both versions can also be accessed through a public GitHub found here and more documentation for the packages can be found here.

Acknowledgements

The development and research of the `FAIRmaterials` package was made possible through generous support from multiple sources. This work was supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the Solar Energy Technologies Office (SETO) through Agreement Numbers DE-EE0009353 and DE-EE0009347. Additional support was provided by the Department of Energy (National Nuclear Security Administration) under Award Number DE-NA0004104 and Contract Number B647887. The authors also gratefully acknowledge funding from the U.S. National Science Foundation under Award Number 2133576.

The authors would like to sincerely thank these organizations for their financial assistance as well as all of the individuals who participated in the project and offered insights and perspectives that were very valuable to the research.

Appendix

Example full set of completed ontology sheets for the mds-XrayToolChess ontology

A	B	C
Prefix Name	**Ontology URL**	**Ontology File**
Create a prefix for the ontology that you would like to import into your own ontology	Enter the URL of the ontology's OWL file	Enter the URI of the location of the ontology OWL online
pmd	https://w3id.org/pmd/co/	
qudt	http://qudt.org/2.1/vocab/unit#	

Figure 4: Namespace Sheet: This sheet is used to define the namespace which connects ontology prefixes to the ontology URL. This sheet aids in preventing conflicts and maintaining clarity across the ontology's vocabulary.

A	B
Ontology Name *Name of the ontology*	mds-XraySample
Ontology URI *Base URI of the ontology (the ontology URI, minus the name of the ontology)*	https://cwrusdle.bitbucket.io/xraySample#
Ontology Version *Version of the ontology*	0.2
Ontology Author(s) *Authors of the ontoloy (seperate multiple authors by a comma)*	Alexander C. Harding Bradley, Balashanmuga Priyan Rajamohan, Mohommad Redad Mehdi, Weiqi Yue, Finley Holt, Pawan K. Tripathi, Erika I. Barcelos, Matthew Willard, Frank Ernst, Roger H. French
Ontology Description *Description of the ontology and ontology domain*	XRD Sample Ontology for the FAST Beamline at CHESS.

Figure 5: Ontology Info Sheet: Contains essential metadata about the ontology including title, creator, and version. This sheet sets the foundational attributes that describe and contextualize the ontology.

A	B	C	D	E	F	G
ValueType Name	**Belongs to Ontology**	**Domain**	**Range**	**Definition of Property**	**Logical Axioms**	**Alternative Name(s)**
The name of the valuetype of a variable that you would like to include in your ontology schema.	If the valuetype already exists in an ontology below, select the ontology from the dropdown menu. Otherwise, please leave blank.	Please enter the term that the relationship starts from. (Example: If you would like to define the relationship that a tool term "outputs" an image term, select the tool term).	Please enter the value type you would like to attach to the term ***Only fill out for Data Property relationships***	Please provide the definition the relationship. For reccomendation on a definition, search the term in https://schema.org. If the term is already in a selected ontology, please leave blank.	Please provide any logic you would like to attach to your term. For information and examples on logical axioms, please visit https://www.w3.org/TR/daml+oil-axioms/ ***Optional***	Please provide any alternative names for the relationship in the literature. ***Optional***
value	pmd	SampleID	xsd:string			
value	pmd	ProcessingMethod	xsd:string			
value	pmd	MaterialName	xsd:string			
value	pmd	**pmd:Length**	**xsd:float**			
value	pmd	**pmd:Width**	**xsd:float**			
value	pmd	**pmd:Thickness**	**xsd:float**			

Figure 6: Value Type Sheet: Specifies the types of values associated with ontology terms, used for data consistency and semantic accuracy in ontology modeling.

A	B	C	D	E	F	G
Relationship Name	**Belongs to Ontology**	**Domain**	**Range**	**Definition**	**Logical Axioms**	**Alternative Name(s)**
The name of the relationship between two terms that you would like to use in your ontology schema.	If the relationship already exists in an ontology below, select the ontology from the dropdown menu. Otherwise, please leave blank.	Please enter the term that the relationship starts from. (Example: If you would like to define the relationship that a tool term "outputs" an image term, select the tool term).	Please enter the term that the relationship goes to. Example: If you would like to define the relationship that a tool term "outputs" an image, select the image variable.	Please provide the definition the relationship. For reccomendation on a definition, search the term in https://schema.org. If the term is already in a selected ontology, please leave blank.	Please provide any logic you would like to attach to your term. For information and examples on logical axioms, please visit https://www.w3.org/TR/daml+oil-axioms/ ***Optional***	Please provide any alternative names for the relationship in the literature. ***Optional***
hasIdentifier	**pmd**	**XraySample**	**SampleID**			
characteristicOf	**pmd**	**XraySample**	**ProcessingMethod**			
characteristicOf	**pmd**	**XraySample**	**MaterialName**			
characteristicOf	**pmd**	**XraySample**	**pmd:Length**			
characteristicOf	**pmd**	**XraySample**	**pmd:Width**			
characteristicOf	**pmd**	**XraySample**	**pmd:Thickness**			
inputOf	**pmd**	**XraySample**	**XrayTool**			

Figure 7: Relationship Definition Sheet: Outlines the various relationships between terms within the ontology, facilitating a structured approach to defining how ontology elements interconnect.

A	B	C	D	E	F	G
Variable Name	**Belongs to Ontology**	**Parent Variable**	**Definition of Variable**	**Alternative Name(s)**	**Unit**	**Logical Axioms**
The variable that you would like to represent in your ontology schema.	If the variable exists in an ontology below, select the ontology from the dropdown menu. Otherwise, please leave blank	Please type the variable from the Parent Ontology that you would like to connect your variable to. If the variable is already in a selected ontology, please leave blank	Please provide the definition the variable. For reccomendation on a definition, search the term in https://schema.org. If the variable is already in a selected ontology, please leave blank.	Please provide any alternative names for the variable in the literature. If the variable is already in a selected ontology, please leave blank ***Optional***	Please provide the unit that the variable is expressed in your data. For a dictionary of standardized units, please visit https://qudt.org. If the variable is already in a selected ontology, please leave blank ***Optional***	Please provide any logic you would like to attach to your variable. For information and examples on logical axioms, please visit https://www.w3.org/TR/daml+oil-axioms/ If the variable is already in a selected ontology, please leave blank
Sample	pmd					
SpecimenName	**pmd**					
SampleID		**pmd:SpecimenName**	**A human-labeled sample identifier.**			
ManufacturingProcess	**pmd**					
ProcessingMethod		pmd:ManufacturingP…	The manufacturing method by ehich the sample was created.			
MaterialDesignation	**pmd**					
MaterialName		pmd:MaterialDesigna…	Name of the material.			
Length	pmd		**Length of the sample.**		qudt:MilliM	
Width	pmd		**Width of the sample.**		qudt:MilliM	
Thickness	**pmd**		Thickness of the sample.		qudt:MilliM	
XraySample		**pmd:Sample**				
XrayTool						

Figure 8: Variable Definition Sheet: This sheet details the individual variables within the ontology, defining their attributes and how they relate to the ontology's broader structure.

References

Carl Boettiger, Anna Krystalli, Bryce Mecum. n.d. "rdflib: Tools to Manipulate and Query Semantic Data." https://cran.r-project.org/web/packages/rdflib/index.html.

Graphviz Development Team. n.d. "Graphviz Python Package." https://pypi.org/project/graphviz/.

Iannone, Rich, and Olivier Roy. n.d. "DiagrammeR: Graph/Network Visualization." https://cran.r-project.org/web/packages/DiagrammeR/index.html.

Lohmann, Steffen, Vincent Link, Eduard Marbach, and Stefan Negru. 2015. "WebVOWL: Web-Based Visualization of Ontologies." In *Knowledge Engineering and Knowledge Management*, edited by Patrick Lambrix, Eero Hyvönen, Eva Blomqvist, Valentina Presutti, Guilin Qi, Uli Sattler, Ying Ding, and Chiara Ghidini, 154–58. Cham: Springer International Publishing.

Musen, Mark A. 2015. "The Protégé Project: A Look Back and a Look Forward." *AI Matters* 1 (4): 4–12. https://doi.org/10.1145/2757001.2757003.

Python Software Foundation. n.d. "PyPI: The Python Package Index." https://pypi.org.

R Project. n.d. "CRAN: The Comprehensive R Archive Network." https://cran.r-project.org.

RDFLib Development Team. n.d. "RDFLib: Python Library for working with RDF." https://pypi.org/project/rdflib/.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (1): 160018. https://doi.org/10.1038/sdata.2016.18.