

GOTTFRIED WILHELM LEIBNIZ UNIVERSITÄT HANNOVER
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK

The ORKG R Package and Its Use in Data Science

*A thesis submitted in fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science*

BY

Zied Boubakri

Matriculation number: 10032060

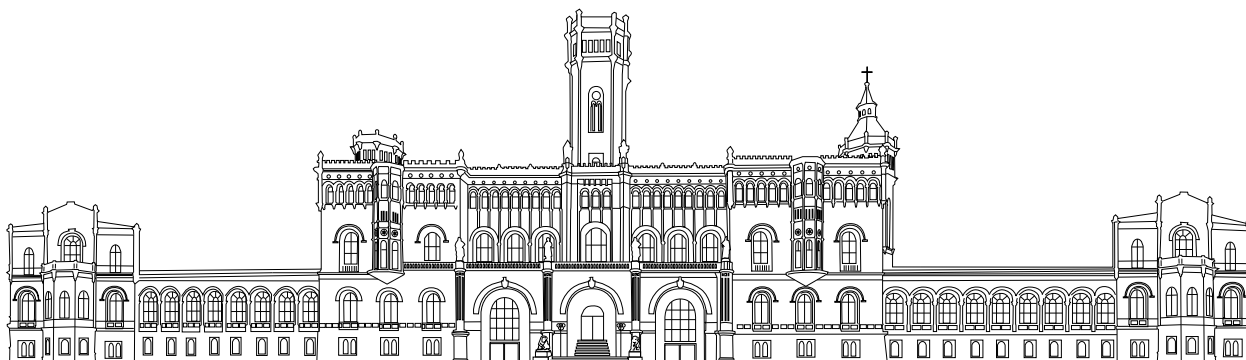
E-mail: zied.boubakri@stud.uni-hannover.de

First evaluator: Prof. Dr. Sören Auer

Second evaluator: Dr. Markus Stocker

Supervisor: Dr. Markus Stocker

28 November 2022



Declaration of Authorship

I, Zied Boubakri, declare that this thesis titled, 'The ORKG R Package and Its Use in Data Science' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Zied Boubakri

Signature: _____



Date: _____

28.11.2022

Inspirational quote

“[...] scientific writing can somewhat cynically be called information burying.”

— Mons, Barend, Article *Which gene did you mean?* [20]

“Information is just bits of data. Knowledge is putting them together[.]”

— Dass, Ram, Book *“One-Liners: A Mini-Manual for a Spiritual Life”* (2007), p.53

Acknowledgements

I would like to begin by expressing my deepest appreciation to my research supervisor, Dr. Markus Stocker. This paper could not have been completed without his direction and unwavering participation at every stage of the process. I would like to express my gratitude for your understanding and support throughout this process.

Prof. Dr. Sören Auer is to be sincerely thanked for giving me the platform to write my bachelor's thesis in the department of Data Science and Digital Libraries.

I'd like to thank Yaser Jaradeh for providing me with a second opinion on technical matters, and I'd also like to thank the entire Department faculty for being so welcoming when I arrived at the office.

My heartfelt gratitude goes to my family for making it possible for me to pursue my education and expand my horizons far from home.

Abstract

Research infrastructures and services provide access to (meta)data via user interfaces and APIs. The more advanced services also support access through (Python, R, etc.) packages that users can use in computational environments. For scientific information as a particular kind of research data, the Open Research Knowledge Graph (ORKG) is an example of an advanced service that also supports accessing data from Python scripts. Since many research communities use R as the statistical language of choice, we have developed the ORKG R package to support accessing and processing ORKG data directly from R scripts. Inspired by the Python library, the ORKG R package supports a comparable set of features through a similar programmatic interface. Having developed the ORKG R package, we demonstrate its use in various applications grounded in life science and soil science research fields. As an additional key contribution of this work, we show how the ORKG R package can be used in combination with ORKG templates to support the pre-publication production and publication of machine readable scientific information, during the data analysis phase of the research life cycle and directly in the scripts that produce scientific information. This new mode of machine readable scientific information production complements the post-publication Crowdsourcing-based manual and NLP-based automated approaches with the major advantages of unmatched high accuracy and fine granularity.

Keywords: ORKG, machine-readable, structured, scholarly knowledge, R-Package, pre-publication

Zusammenfassung

Forschungsinfrastrukturen und -dienste ermöglichen den Zugang zu (Meta-)Daten über Benutzerschnittstellen und APIs. Die fortgeschritteneren Dienste unterstützen auch den Zugang über (Python-, R- usw.) Pakete, die die Nutzer in Rechenumgebungen verwenden können. Für wissenschaftliche Informationen als eine besondere Art von Forschungsdaten ist der Open Research Knowledge Graph (ORKG) ein Beispiel für einen fortgeschrittenen Dienst, der auch den Zugriff auf Daten über Python-Skripte unterstützt. Da viele Forschungsgemeinschaften R als statistische Sprache der Wahl verwenden, haben wir das ORKG R-Paket entwickelt, um den Zugriff auf ORKG-Daten und deren Verarbeitung direkt aus R-Skripten zu unterstützen. Inspiriert von der Python-Bibliothek, unterstützt das ORKG R-Paket eine vergleichbare Reihe von Funktionen durch eine ähnliche programmatische Schnittstelle. Nachdem wir das ORKG R-Paket entwickelt haben, demonstrieren wir seine Verwendung in verschiedenen Anwendungen in den Bereichen Biowissenschaften und Bodenforschung. Als weiteren wichtigen Beitrag dieser Arbeit zeigen wir, wie das ORKG R-Paket in Kombination mit ORKG-Vorlagen verwendet werden kann, um die Erstellung und Veröffentlichung maschinenlesbarer wissenschaftlicher Informationen vor der Veröffentlichung zu unterstützen, und zwar während der Datenanalysephase des Forschungslebenszyklus und direkt in den Skripten, die wissenschaftliche Informationen produzieren. Diese neue Art der Erstellung maschinenlesbarer wissenschaftlicher Informationen ergänzt die auf Crowdsourcing basierenden manuellen und NLP-basierten automatisierten Ansätze nach der Veröffentlichung mit den großen Vorteilen einer unübertroffenen hohen Genauigkeit und feinen Granularität.

Stichworte: ORKG, maschinenlesbar, strukturiert, wissenschaftliches Wissen, R-Paket, Vorveröffentlichung

Contents

1	Introduction	1
1.1	Problem statement	2
1.2	Research questions	3
1.3	Software availability statement	3
1.4	Structure	3
2	Related Work	5
2.1	Scholarly information infrastructures	5
2.1.1	Infrastructures for metadata	5
2.1.2	Infrastructures for scientific information	7
2.2	Data repositories	8
2.3	Methods	9
2.3.1	Manual extraction	9
2.3.2	Automated extraction	10
3	Approach	12
3.1	Architecture	12
3.2	Implementation	13
3.2.1	Overall system	14
3.2.2	ORKG REST API	15
3.2.3	R/Python Package	15
3.2.4	ORKG templates	16
3.2.5	Publishing and harvesting machine-readable information	18
4	Results	20
4.1	ORKG R package	20
4.2	Retrieving ORKG content	21
4.3	Retrieving tabular data as R data.frame	22
4.4	ORKG Comparisons	23

4.5	Materializing templates	24
4.6	Using templates and serializing data	25
4.7	The conversion from R's Dataframe to annotated table	27
4.8	Tabular data display in ORKG	29
5	Applications	30
5.1	Life science application	30
5.2	Soil science application	36
6	Discussion	42
6.1	Discussion of research questions	42
6.2	Advantages of the proposed approach	43
6.3	Limitations of the work	43
6.4	Future work	44
7	Conclusion	45
	Bibliography	47

List of Figures

3.1	Scholarly knowledge graph-driven platform with researchers, publishers, and a distributed system, as well as their connection via intermediate components, as the four primary elements to be interpreted in the system architecture.	13
3.2	This more thorough design focuses on ORKG as the infrastructure for scholarly knowledge, as well as the template approach and serialization into JSON-LD, to which the DOI will connect alongside the metadata of its paper.	14
3.3	It can be inferred from the p-value template description that the OBI ontology term “OBI_0000175” is the target class of this template.	17
3.4	The one property of the p-value template is the value specification.	18
4.1	A portion of what the method <code>get</code> produces as an output, in the case of resources.	21
4.2	This is the targeted resource.	22
4.3	This is how it is displayed in ORKG.	22
4.4	This is the resulting Dataframe after executing the snippet of code in the Listing 4.3 in the Jupyter Notebook.	23
4.5	The matrix displayed after retrieving the comparison with the ID R196200.	24
4.6	The documentation of the template and the serialization process.	26
5.1	The targeted Dataframe from the research paper.	31
5.2	Metadata after looking up the DOI link.	33
5.3	Properties filled automatically, after setting the research field and a directional arrow pointing to the visualization button.	34
5.4	The tabular visualization of the life science use case, after clicking pressing the visualization button.	35
5.5	Here is a sample of code that displays the table in a boxplot after retrieving it from ORKG.	35
5.6	The boxplot represents a summary of the data shown in the ORKG table.	36
5.7	The Dataframe used for the first contribution.	36

5.8	The Dataframe used for the second contribution.	37
5.9	The doc-string of the templates used in the soil science use case.	37
5.10	The exhibition of the soil science use case and the button that lead to the tabular visualization.	40
5.11	The tabular visualization of the second Dataframe presented in the soil science use case in ORKG.	40
5.12	In this snippet of code, the ORKG table was retrieved and then plotted in three bar graphs.	41
5.13	The source of the three bar graphs was the table retrieved from ORKG. . .	41

List of Tables

2.1	Platforms that provide information extracted from papers.	7
-----	---	---

Acronyms

API Application Programming Interface

CoDa Cooperation Databank

COVID-AQS COVID-19 Air Quality Data Collection

CRAN Comprehensive R Archive Network

CSV Comma-Separated Values

CSVW Comma-Separated Values on the Web

DOI Digital Object Identifier

FAIR Findable. Accessible. Interoperable. Reusable.

JSON-LD JavaScript Object Notation for Linked Data

NLP Natural Language Processing

ORCID Open Researcher and Contributor iD

ORKG Open Research Knowledge Graph

PID Persistent Identifier

RDF Resource Description Framework

RI Research Infrastructure

ROR The Research Organization Registry

SN SciGraph Springer Nature SciGraph

SKG Scholarly Knowledge Graph

SPARQL SPARQL Protocol and RDF Query Language

URI Universal Resource Identifier

W3C World Wide Web Consortium

XML Extensible Markup Language

Chapter 1

Introduction

A knowledge graph can be defined as a huge network of interconnected real-world entities along with their semantic types and properties ([5], [15], [22], [24]). Knowledge graphs make it easier to aggregate heterogeneous data from various sources, even if they are of varying quality, into interlinked data that is more manageable and can be shared and reused effectively. This data enables users to see subtler patterns that were previously concealed. Knowledge graphs also bring other advantages, such as eliminating the need to sift through mounds of paperwork to locate the one document one needs by displaying just the most relevant information. Moreover, Knowledge Graphs are the gold standard for depicting complex networks due to their capacity to rapidly and precisely describe relationships and the fact that they are highly customizable.

The FAIR principles are guidelines to ensure that (meta)data is findable, accessible, interoperable, and reusable. A number of requirements must be satisfied for data to be readily discoverable, including having a persistent identifier (such as a DOI or URL), being described by rich metadata that complies to established standards, and being featured on regional and national data discovery portals. Next, data is considered accessible if it can be obtained by humans and machines with little effort and according to well-defined, ideally standardized protocols. With interoperability, it is possible to merge diverse kinds of data, for instance in the same web application. This is ensured by the use of a standard programming language, a constrained vocabulary, and open file formats. It is essential that both data and metadata comply to standards to enhance its utility for later applications (reusability). Both the data and the metadata, as well as the use licenses and any other information relevant to the area of interest for the data, should be stated in detail[30].

Knowledge graphs and the FAIR principles were introduced to demonstrate how a Knowledge Graph-driven FAIR scientific information platform like ORKG¹ can be a game changer, how this next-generation digital infrastructure [13] can reach its full potential with more efficient ways to publish accurate and granular scientific information buried in scholarly articles in structured manner with pre-publication approaches that ensure scientific information is produced FAIR. These approaches have considerable advantages over post-publication Crowdsourcing-based manual extraction or NLP-based automated extraction techniques, specifically: technological simplicity, high accuracy and quality, and unmatched granularity of information.

1.1 Problem statement

The problem at hand is the production of FAIR scientific information and the problem known as the “data acquisition bottleneck.” The conversion of the vast amounts of scientific information in PDF files into machine readable formats is a major bottleneck in the data acquisition process, for a variety of technical as well as human factors, since it remains unclear how users will produce FAIR scientific information and the role of machines in this process. Researchers are unlikely willing to ensure scientific information is machine readable, which is an important factor in the data acquisition bottleneck. The lack of structured scientific information is a big impediment to the efficiency of science, since the reuse of scientific information is expensive and relies on extracting, at high cost, information previously buried in text [20].

Key objective of this work is to create solutions that ensure scientific information is produced machine readable when information is created in data analysis. The intuition is that during the data analysis the data consumed and produced is in structured form, meaning that all the necessary data, and more specifically scientific information ultimately published in papers, is available and processable. It is thus far cheaper and more accurate to ensure scientific information is produced in machine readable form in this manner as opposed to manually or automatically extracting information from articles at a later stage. Post-publication extraction is expensive (e.g. in building extraction models or performing extraction manually), inaccurate (especially NLP-based extraction, but also manual extraction is prone to errors), and is unable to extract rich information with high granularity.

¹<https://orkg.org/>

1.2 Research questions

RQ1) How can we ensure that scientific information is produced machine readable?

This is in contrast to more traditional post-publication approaches, where researchers write papers and semi-automated processes then extract information from papers to create structured information. To address this question, our approach integrates machine-readable information production into data analysis. For this, we develop the ORKG R package to support the ORKG template-based production of machine readable scientific information, which is then published and harvested, specifically by ORKG.

RQ2) How to dynamically generate programmatic interfaces in R based on ORKG template?

To guide users in the production of machine readable scientific information in R scripts, it is beneficial if the ORKG R package provides programmatic interfaces that reflects the specification of ORKG templates. Since ORKG templates can be created or modified, the interface needs to be created dynamically. This avoids the need to modify and redeploy the ORKG R package in local computational environments.

RQ3) How to ensure that machine readable scientific information is published and can be harvested, given article DOIs?

Linking machine readable scientific information to articles is an important step so that systems such as ORKG can easily discover and harvest such content.

1.3 Software availability statement

During the course of this thesis, an ORKG R package was implemented, and the release v1.0.0² is the final version before the thesis is submitted.

1.4 Structure

The thesis is structured as follows. Chapter 2 addresses the related work briefly and puts this study in perspective. Chapter 3 introduces the architecture in general terms, followed by a more technologically detailed presentation of the essential components. Chapter 4 presents the findings of the study as well as the implementation, which includes the ORKG R package. Chapter 5 discusses the application and

²<https://gitlab.com/TIBHannover/orkg/orkg-r/-/releases/v1.0.0>

shows two use cases, one in life science and one in soil science. Chapter 6 discusses the findings and their application, highlighting the benefits, limitations, and some future work. We provide concluding remarks in Chapter 7.

Chapter 2

Related Work

This chapter presents work related to this thesis. We discuss methods related to machine readable scientific information production (especially NLP and crowdsourcing based approaches) as well as research infrastructures that manage, curate, and publish scientific information (both bibliographic metadata and data).

This thesis is not the first initiative focusing on efficient access and reuse of research data in computational environments. Many research infrastructures or data repositories, such as the NEON¹ project, have identified the problem and developed approaches. In contrast to these, our focus on making scientific information machine readable at production in computational environments is, however, novel.

2.1 Scholarly information infrastructures

2.1.1 Infrastructures for metadata

With the **OpenAIRE Research Graph**², funders, organizations, researchers, research communities, and publishers have access to a semantic graph database that collects a range of research data attributes (metadata, linkages) from the OpenAIRE Open Science infrastructure [19]. Information on research life-cycle objects is gathered via the OpenAIRE technological infrastructure from a wide variety of sources, including scholarly journals, software metadata, dataset metadata from data repositories, and full-text data journals. OpenAIRE’s internal metadata model is applied to collected metadata to produce the OpenAIRE Research Graph, which is available through the OpenAIRE site and APIs [3].

¹<https://www.neonscience.org/>

²<https://graph.openaire.eu>

The **PID Graph**³ provides access to heterogeneous PID Systems, including DataCite⁴, Crossref⁵, ORCID⁶ and ROR⁷. Persistent Identifiers guarantee that entities are referred to consistently through time and metadata about the identified entities is persistently available, even if the entity itself is not. PID Graph increases the value of PIDs by linking them via metadata not just about the identified resources themselves, but also about their connections, uniting varied entities in the research environment and providing researchers and academic institutions with access to new information[6, 9]. The PID Graph is powered by a GraphQL-based API operated DataCite, Users need to parse the JSON responses supplied by the API in order to export the data into a computational environment [8].

The **Springer Nature SciGraph**⁸ offers a comprehensive depiction of the relationship between various interconnected datasets, hence enhancing the discoverability of Springer Nature papers and enabling researchers and developers to overcome data silos through data access. It is designed for both developers and linked data specialists, who can locate machine-readable representations of SN SciGraph resources by using the Linked Data API to retrieve RDF, downloading the published bulk data, or accessing JSON-LD using REST via the Springer Nature Meta API. With the SN SciGraph Explorer providing an overview of the variety of available data and how the data is structured SciGraph also caters the general research community [28].

OpenAlex⁹ is an open data, API, and source code repository for scholarly metadata and an open scientific knowledge graph. The OpenAlex dataset is made up of five different kinds of academic entities and the relationships between them, forming a heterogeneous directed network. Works are academic writings such as articles, books, datasets, and theses; authors are the people who produce these works; venues are the physical locations where these works are presented; institutions are the organizations to which authors claim affiliations; and concepts are the overarching ideas that these works explore. These entities may be located in a web-based graphical user interface (GUI) or in a downloadable data dump, obtained from a REST API as a JSON object, all of which use the same permanent OpenAlex ID as the main key in the dataset [23].

³<https://www.project-freya.eu/en/pid-graph/the-pid-graph>

⁴<https://datacite.org/>

⁵<https://www.crossref.org/>

⁶<https://orcid.org/>

⁷<https://ror.org/>

⁸<https://www.springernature.com/de/researchers/scigraph>

⁹<https://openalex.org/>

2.1. Scholarly information infrastructures

	Full Name	URL	Field
Hi-Knowledge	Hi-Knowledge	https://hi-knowledge.org/	Invasion Biology (Science and nature)
CoDa	Cooperation Databank	https://cooperationdatabank.org/	Cooperation in social dilemmas
Plazi	Plazi	https://plazi.org/	Biodiversity (Taxonomic literature)
COVID-AQS	COVID-19 Air Quality Data Collection	https://covid-aqs.fz-juelich.de/	The impacts of COVID-19 lockdowns on air quality
Papers With Code	Papers With Code	https://paperswithcode.com/	Machine learning/AI

Table 2.1: Platforms that provide information extracted from papers.

2.1.2 Infrastructures for scientific information

The scholarly information infrastructures for metadata presented so far focus on identifying and describing entities, such as articles, datasets, and people. The assets themselves are, from the point of view of these systems, binary objects. Specifically, the scientific information expressed in articles is not made accessible and processable by these systems. There exist many community-driven infrastructures and services supporting the publishing and reuse of machine readable scientific information. These services are typically tailored for specific research fields and even research questions. Except ORKG, which serves as the infrastructure for this thesis, the following infrastructures are summarised in the table 2.1.

Hi-Knowledge¹⁰ is an example in biodiversity and supports the visualization of relationships between hypothesis in invasion biology, and their support or questioning in the literature. The presented data is scientific information extracted from hundreds of published articles. The services provides the data as downloadable CSV files.

The **Cooperation Databank**¹¹ is an example in social science with scientific information about human cooperation extracted from approx. 2000 published articles [29]. Being a more advanced infrastructure, the Cooperation Databank is powered by a triple store database with a SPARQL endpoint that supports flexible querying of content. While such an endpoint makes it easy to fetch data, users still don't get directly a language-specific data structure such as a DataFrame in Python or R. Rather, users need to iterate over the results provided in XML.

Plazi¹² is an organization that helps to preserve and make accessible to the public published works in the context of digital taxonomy. Its purpose is to define data useful for comprehending global biodiversity and to construct a catalog of all

¹⁰<https://hi-knowledge.org/>

¹¹<https://cooperationdatabank.org/data/>

¹²<https://plazi.org/>

known species on Earth. This taxonomic literature is transformed into semantically enhanced XML documents that may be accessed [1].

The **COVID-19 Air Quality Data Collection** (COVID-AQS)¹³ publishes the scientific information extracted from approximately 150 articles, that address the impacts of COVID-19 lockdowns on air quality, through a downloadable CSV dataset that is freely and openly accessible for scientific use. Using this data, the service provides filtering and visualizations of the data, including by Country/Region. [10].

Papers With Code¹⁴ is a knowledge-sharing platform that collects algorithms-based machine learning research papers of authors or communities in the field of artificial intelligence in a manner that all of their summary, code implementation, result, and data are extracted for scientific articles and presented by innovative services visualizing the trends of algorithm performance for machine learning tasks.

The **Open Research Knowledge Graph** (ORKG) is being developed as an Open Source project by the German National Library of Science and Technology (TIB) with the intention of expressing scientific information published in articles in machine readable form as a knowledge graph rather than a traditional set of documents [13]. By organizing scientific information in a knowledge graph, services can more easily support users in searching, comparing, visualizing and reusing scientific information. In addition, ORKG provides an ORKG Python package¹⁵, which was initially developed to support adding, changing, and reading ORKG content, including loading ORKG comparison data into data analysis environment such as Jupyter notebooks, to support subsequent data analysis to easily produce advanced visualizations or data products, as well as execute complex data-enabled activities that integrate ORKG data with other data analysis tools [4, 7].

2.2 Data repositories

Numerous data repositories provide tailored libraries in Python or R that support the direct access to data and loading into computational environment, making thus easier to reuse, process, integrate and visualize published data [12]. Examples are NEON and PANGAEA¹⁶.

PANGAEA has existed as a repository offering access to georeferenced data from the Earth, environmental, and biodiversity sciences for nearly three decades. R and

¹³<https://covid-aqs.fz-juelich.de/>

¹⁴<https://paperswithcode.com/>

¹⁵<https://orkg.readthedocs.io/en/latest/introduction.html>

¹⁶<https://www.pangaea.de>

Python packages are given to communicate with the PANGAEA database. Pangear is a R package that provides an interface for retrieving data from PANGAEA. It facilitates metadata queries and the download of tabular PANGAEA datasets. Pangaeapy is a Python package used to obtain and analyze metadata and data from the tabular PANGAEA dataset[12, 26].

The National Ecological Observatory Network is a distributed site-based Research Infrastructure (RI) of observatories throughout the continent that will gather and transmit ecological data on the consequences of climate change, land use change, and invasive species during the next three decades[12, 25]. Through the NEON Data Portal or API, R and Python packages (NEON-utilities) can access NEON data. The Data Portal provides monthly zip files of data, whereas the API provides individual files. These packages help find, download, and format data for analysis. This covers API data downloads, table merges, and format conversions.

However, the data accessed from these two data repositories through such libraries is typically primary research data, such as observational or simulation data, rather than scientific information. In other words, the access data is not extracted from the articles.

2.3 Methods

2.3.1 Manual extraction

Crowdsourcing is a helpful research approach since it enlists the general public's participation in reading research articles, even if they are not professionally qualified specialists, and displaying the results of that study. There is no denying that the capacity to gather or analyze data on a much larger scale is the most significant advantage of crowdsourcing. Gains in speed, throughput, and cost are possible due to the presence of many people. If researchers want to employ crowdsourcing, they may do it by using a variety of resources already available online. Researchers may use these resources to crowdsource tasks like data collection, image classification, systematic reviews, new ideas, and financing, but there are caveats to keep in mind and biases to account for [18].

In **systematic reviews**, the unit of interest are articles. Hence, it is required to identify and integrate results from typically hundreds of related articles. It is essential to perform exhaustive research, the quality of which is dependent on the use of suitable scientific review methodologies. Systematic reviews gather and assemble information on relevant studies, including their methodologies, possible bias, and conclusions, in order to address research questions. Therefore, the presentation and

analysis of data from these studies has a substantial effect on the conclusions of a systematic review. Since systematic reviews are retroactive research, they are not immune to bias. Because of this, time-consuming systematic reviews may be too deceiving, ineffective, or even dangerous if data are improperly managed [17, 31].

2.3.2 Automated extraction

NLP is a technique used by ORKG for automatic extraction. The goal of **natural language processing (NLP)** is to convert unstructured content into a machine-readable representation so that a computer system can use it to carry out a number of tasks. This is accomplished by employing computational methods to evaluate texts and by researching how individuals comprehend and use language. Common NLP tasks include: Named Entity Recognition, Entity Linking, Topic Modeling, Text Summarization. The primary objective of Named Entity Recognition (NER) is identifying and classifying important information (called entities) from sentences, paragraphs, text reports, and other unstructured text formats into specified categories [16]. Entity Linking, on the other hand, aims to provide context to data by providing annotations for ideas, concept instances, and relations between the two [27]. Third, Topic Modeling is an unsupervised method for extracting information from textual materials and displaying it as categories [2]. Finally, Text Summarization is the act of condensing a large piece of literature into a shorter version while preserving its meaning and information value [14]. [21]

We highlight the benefits and drawbacks of pre-publication scholarly knowledge production in comparison to automated extraction using NLP as a tool:

- Given the depth and precision provided by the fact that researchers and writers are typically the ones who explain their work, this method of describing scientific knowledge is relatively affordable and accurate at a level that NLP extraction would rarely accomplish.
- Compare this to the expensive and time-consuming process of training models in natural language processing; in ORKG, the researchers and authors' biggest challenge will be choosing the right structure to best represent their paper; everything else is easy once they have the script down and understand how it works and have their input data ready.
- However, this strategy is not readily scalable because to the difficulty of persuading and pressuring academics and writers to participate. This is the battle where natural language processing (NLP) may be most beneficial. If it achieves

a high degree of accuracy and can be automated, hundreds of papers may be processed in a couple of hours, but our technique based on persuasion may need more time to achieve shared aims.

Chapter 3

Approach

This chapter explains the architecture and the technical specifics of the proposed implementation. We thus distinguish the architecture of the system, which is abstract and emphasizes the key components and their interactions, from the more detailed technical description of a proposed implementation, which includes all of the used components and their connections.

3.1 Architecture

As shown in Figure 3.1, it all starts with a research infrastructure. Data from this infrastructure should be accessible to researchers for analysis or as guidelines for making unstructured data machine-readable. This research infrastructure should offer a possible entry point. For example, a library in a given language allows the researcher to access this information in a specific computational environment. Once the research manuscript has been published with the retrieved and processed data and some new results, this information is often only available in an unstructured form (text) in the research paper. A second researcher would then resort to data mining to obtain the same information. In light of this, the architecture significantly encouraged the creation of data that could be reused. This information has to be stored in a machine-readable manner. To save this last, the researcher would have to submit the acquired data as additional material alongside the paper to the publication. The publishing of all submitted content would result in a DOI that would offer the metadata of the article as normal, but would also include a new link to this supplementary material. This link should subsequently be discoverable by distributed systems, which will display this research on their platform, utilizing both the traced machine-readable data and its metadata.

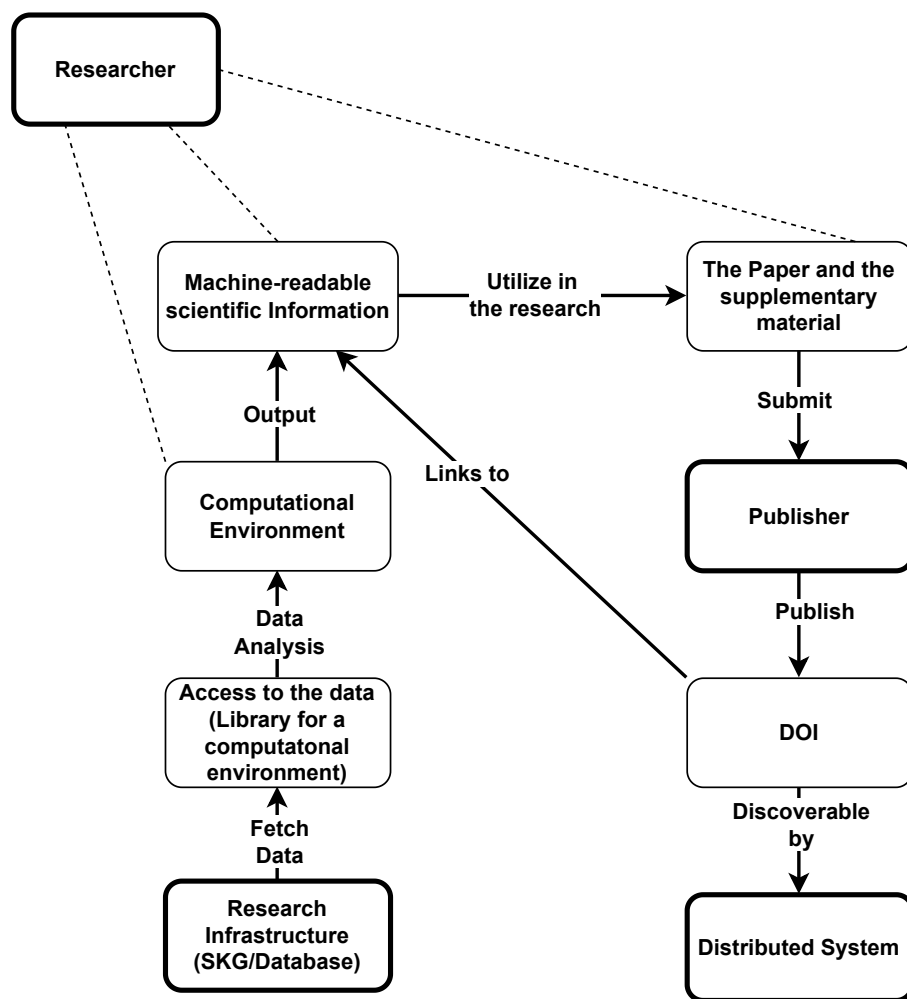


Figure 3.1: Scholarly knowledge graph-driven platform with researchers, publishers, and a distributed system, as well as their connection via intermediate components, as the four primary elements to be interpreted in the system architecture.

3.2 Implementation

Initially, The implementation will be described as a whole. Afterward, the description of some of the primary components, which may not be immediately apparent, and some of their relationships will be presented.

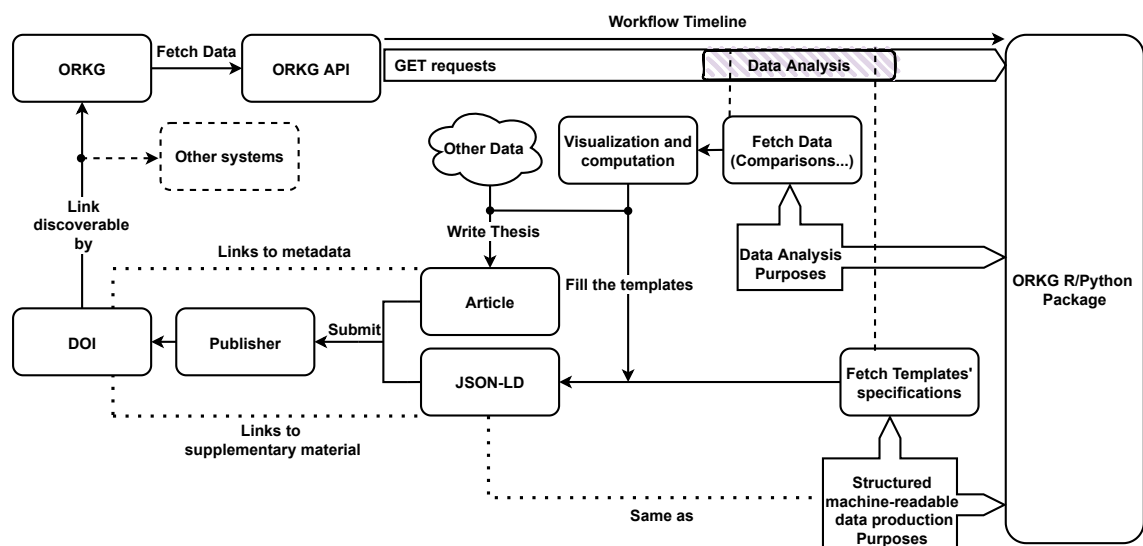


Figure 3.2: This more thorough design focuses on ORKG as the infrastructure for scholarly knowledge, as well as the template approach and serialization into JSON-LD, to which the DOI will connect alongside the metadata of its paper.

3.2.1 Overall system

The implementation will be outlined according to three phases of the research life cycle: early data analysis phase, later data analysis phase, and publication phase. During the early phase of data analysis, researchers collect data from various sources. ORKG may be such a source, specifically for scientific information. Researchers may utilize ORKG libraries in Python or R to obtain data from ORKG into a data structure specific to the computing environment in which the analysis is being conducted. Following loading data, the data is ready to be processed, visualized, used in the manuscript or stored for the next phase of the research life cycle. The next step in the research life cycle is for the researcher to compile all the gathered data and further organize it by using it to fill the templates, the specifications of which are to be retrieved from ORKG. These templates aid in the creation of machine-readable data by making it apparent what information the template needs from its user and by streamlining the process of serializing data into JSON-LD, the recommended supplementary format. When it comes time for publishing, this machine-readable data will be submitted alongside the manuscript to the publisher. Following that, a DOI will be made public. By processing the DOI metadata, both the metadata of the article and the supplementary material should be traceable, at least by ORKG, given

that this feature is already available, and maybe by other systems if this approach is more often adopted.

3.2.2 ORKG REST API

ORKG provides a REST API¹ to access the raw data. It is essential to note that this is not the data format expected in a computing environment, but rather the source format of the scientific information retrieved from ORKG.

Because we are more concerned with retrieving information from ORKG rather than making changes to it, GET is the only HTTP verb from the API that are taken into consideration here.

When considering the purpose of the fetching, GET requests are divided into two categories: the first is for retrieving data that can be used for research purposes. For instance, computing mean of basic reproduction number given some retrieved inputs or directly retrieving this and visualizing it. The second purpose is for retrieving the specification of templates, which support the production of machine-readable scientific information in the computational environment using the Python or R package.

The Figure 3.2 shows these two connections to the ORKG API, the first in the workflow the read of data, which may be utilized in the early stages of data analysis phase, and then second the fetch template specification to be filled to generate machine-readable structured scientific information.

Suppose both of these different GET Requests from the ORKG API were to occur in the workflow since the first type is unnecessary. In that case, they should happen at two different time points, so the first one, where data gets fetched from ORKG for comparisons, occurs much earlier in the workflow. Data analysis requires that all of the data is already gathered. However, fetching the templates needed for producing structured scientific information is probably the last step of data analysis. Nonetheless, they are both in the pre-publication phase.

3.2.3 R/Python Package

It may be useful to begin with an explanation of why these specific programming languages were selected.

Since Python² became popular, there has been heated discussion on whether R³ is superior than Python or serves a different purpose. However, some users choose

¹<http://tibhannover.gitlab.io/orkg/orkg-backend/api-doc/>

²<https://www.python.org/>

³<https://www.r-project.org/>

to disregard R and avoid using it because of its restrictions, which largely consist of how slowly it works in comparison to Python and its difficult maintenance. Aside from the theoretical interest, none of this should be seen as a reason to always use Python instead of R. Therefore, as long as there are scientists and users in general using R for data analysis, it is useful to support this community in using ORKG as a platform and ORKG data in R computing environments.

Furthermore, seeing how R and Python somewhat complement each other, with R being a good fit for data scientists interested in statistical calculation and data visualization and Python being a better fit for data scientists interested in big data, artificial intelligence, and deep learning algorithms, made that choice even clearer.

The **ORKG Python library**⁴, described in the Chapter 2 is the first initiative to facilitate the interaction between ORKG and a computational environment, allowing users to create and modify data.

To support the R community, we developed the **ORKG R package**⁵. Its implementation and its supported features are inspired by the Python library, particularly in aspects of fetching the data. However, due to differences in Python and R languages, the two packages are not entirely equal, regarding their approach to the same features .

The two packages are essentially binders between the ORKG API and Python or R computing environments. Their features are: (1) loading ORKG comparison data into Dataframes and (2) dynamically create a programmatic interface reflecting one or more ORKG templates, as well as creating template-based data serialized in JSON-LD.

3.2.4 ORKG templates

It is difficult to structure academic scientific content. The extent to which a research contribution must be described, the issues that must be resolved, the conclusions gained, and the resources and approaches used are all aspects that must be considered. It is also critical to characterize research contributions to the same topic consistently. This is why ORKG tackles this issue by introducing a new approach to guidelines called Templates, which resembles schemes. All that is required of the user is to fill these templates with inputs that adhere to their specifications. Following these templates allows the researcher to provide material in a more organised manner. Furthermore, having that specification of each template makes it straight-

⁴<https://gitlab.com/TIBHannover/orkg/orkg-pypi/>

⁵<https://gitlab.com/TIBHannover/orkg/orkg-r>

3.2. Implementation

forward for distributed systems to know how to show this knowledge. Finally, by sharing the same structure, it makes data easier to compare [4].

In ORKG, Format, Properties, and Description are the three sections that make up a template (Figure 3.3). The template’s general information, such as its name and target class, are specified in the Description Tab. The name of the template is used to find it while the target class is what the template’s instances will be based on. Furthermore, users can specify for which research fields and research problems this template is relevant.

p-value

Description Properties Format

Name of template

p-value

Target class (optional)

obo:OBI_0000175

Template use cases

These fields are optional, the property is used to link the contribution resource to the template instance. The research fields/problems are used to suggest this template in the relevant papers.

Property (optional)

No Property

Research fields (optional)

No research fields

Research problems (optional)

No research problem

Figure 3.3: It can be inferred from the p-value template description that the OBI ontology term “OBI_0000175” is the target class of this template.

The properties tab is where we define the fields that users will be asked to fill in, along with their respective allowed values and cardinality. For example, in Figure 3.4, the property “value specification” is an instance of a template, the target class of which is an OBI ontology term. This component can only appear once in the p-value template.

p-value

Description Properties **Format**

Property	Type
value specification	<input type="text" value="obo:OBI_0001931"/>
Cardinality	Custom...
Minimum Occurrence	1
Maximum Occurrence	1

Figure 3.4: The one property of the p-value template is the value specification.

Although the format tabs in the templates are optional, they do allow users to generate a label according to the template’s specifications⁶. The approach is elaborated on in the subsequent chapter 5.

3.2.5 Publishing and harvesting machine-readable information

The collected and resulting data from the data analysis are used to populate specified templates. This template data is then serialized as machine-readable scientific data in JSON-LD⁷. JSON-LD⁸ files are one example of supplementary contents that may be submitted with the article to be published. The JSON-LD standard’s main focus is on making it easier for Linked Data⁹ to be stored in JSON-based systems. After the serialization, the researcher submits both the manuscript and the serialized data as supplementary material to the publisher.

⁶<https://orkg.org/about/19/Templates>

⁷<https://www.w3.org/TR/json-ld11/>

⁸<https://www.w3.org/TR/json-ld11/>

⁹Linked data refers to information that has been made publicly accessible in a predefined format for the purpose of reuse and linking. <https://www.w3.org/standards/semanticweb/data>

When these materials are published, the machine-readable scientific information generated during the data analysis phase is then linked as supplementary material to the paper through DOI metadata by means of the “IsSupplementedBy” relation type for related identifiers. This allows the machine-readable scientific knowledge to be discovered via the article’s DOI, and thus enabling harvesting.

Now, from the perspective of the system, given an article DOI, machines read DOI metadata, follow “IsSupplementedBy” links to discover machine-readable scientific material, and then incorporate this data into its system.

Chapter 4

Results

This chapter presents the ORKG R package, particularly its functionality. Special emphasis is given on using the package to retrieve ORKG comparison data as well as to create ORKG template-based scientific information in data analysis.

4.1 ORKG R package

The ORKG R package also tends to help the data meet more than three of the FAIR principles, especially when it comes to accessibility, interoperability, and reuse. This is explained by the fact that the package allows access to ORKG contents via their IDs and converts these entities to R data structures. Tested in both R Console and Jupyter Notebook, this permits the data to be processed and visualized.

Since the package is not published in CRAN, it is required to have the source code and install this package locally with “devtools” in the user’s console. The Listing illustrates the two steps required to install this package. 4.1.

Listing 4.1: These steps helps install locally the package.

```
devtools::document()  
devtools::install()
```

To initialise the ORKG connector, the package should be imported, and then the instance of ORKG should be initialised with a host, which is of type string. For now a host can either be a local host, <https://orkg.org> or <https://sandbox.orkg.org>. These steps are displayed in the Listing 4.2.

Listing 4.2: Simply by adding these two lines, an ORKG connector will be initialised. Remember to add a host address.

```
library(orkg)
orkg = ORKG(host="<host_address>")
```

4.2 Retrieving ORKG content

The ORKG R package provides access to a variety of ORKG content types, specifically resources, classes, statements, literals, predicates, comparisons, and templates. The latter two types will be discussed in more detail in this chapter.

The first five types listed share almost the same strategy for retrieval. The entity may be retrieved by its ID using the method `by_id`. This returns a list containing label, creation data, and so on. By default, a list of 20 entities can be retrieved too when calling the `get` (or `get_all` in some cases) method with the index "content."

```
In [13]: library(orkg)
orkg <- ORKG(host="https://orkg.org")
orkg$resources$get()$content[1:8,1:4]
```

id	label	classes	shared
R0	Gruber's design of ontologies		1
R172	Oceanography	ResearchField	152
R173	Physics	ResearchField	8
R174	Astronomy and Astrophysics	ResearchField	1
R175	Atomic, Molecular and Optical Physics	ResearchField	132
R176	Biological and Chemical Physics	ResearchField	2
R177	Condensed Matter Physics	ResearchField	1
R178	Cosmology, Relativity, and Gravity	ResearchField	2

Figure 4.1: A portion of what the method `get` produces as an output, in the case of resources.

Only the statements and resources as entities stand out due to their extra retrieval methods, such as `get_by_predicate` or `get_by_object` etc. for the statement, which were advantageous for the display of certain types of resources, which leads to the following result.

4.3 Retrieving tabular data as R data.frame

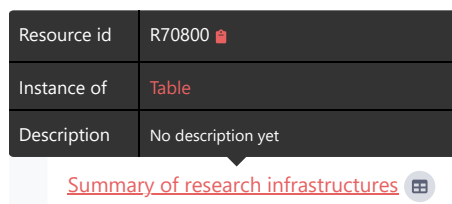
If the retrieved resource is of type `orkg:Table`, the method `as_dataframe()` converts the retrieved tabular data into an R `data.frame`.

When a resource is requested using the method `by_id` that returns the retrieved resource using the ID in a list format, the `as_dataframe` method is appended to the list for reference. This is because the method simply returns the result of this conversion that occurred during the resource retrieval.

As an example of use, in the following code snippet, the test-oriented temporary resource with the ID R70800, which is shown in the Figures 4.2 and 4.3, is fetched. The `as_dataframe` method will gather all of the values and present them in a `Dataframe` format, as seen in Figure 4.4.

Listing 4.3: This is the code snippet for retrieving the resource into a dataframe.

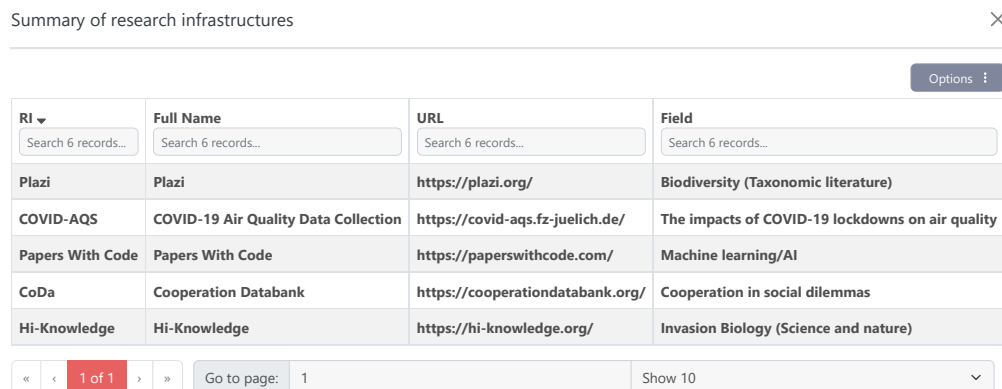
```
orkg = ORKG(host="https://sandbox.orkg.org")
orkg$resources$by_id("R70800")$as_dataframe()
```



Resource id	R70800
Instance of	Table
Description	No description yet

[Summary of research infrastructures](#)

Figure 4.2: This is the targeted resource.



RI	Full Name	URL	Field
Plazi	Plazi	https://plazi.org/	Biodiversity (Taxonomic literature)
COVID-AQS	COVID-19 Air Quality Data Collection	https://covid-aqs.fz-juelich.de/	The impacts of COVID-19 lockdowns on air quality
Papers With Code	Papers With Code	https://paperswithcode.com/	Machine learning/AI
CoDa	Cooperation Databank	https://cooperationdatabank.org/	Cooperation in social dilemmas
Hi-Knowledge	Hi-Knowledge	https://hi-knowledge.org/	Invasion Biology (Science and nature)

« < 1 of 1 > » Go to page: 1 Show 10

Figure 4.3: This is how it is displayed in ORKG.

RI	Full Name	URL	Field
Plazi	Plazi	https://plazi.org/	Biodiversity (Taxonomic literature)
COVID-AQS	COVID-19 Air Quality Data Collection	https://covid-aqs.fz-juelich.de/	The impacts of COVID-19 lockdowns on air quality
Papers With Code	Papers With Code	https://paperswithcode.com/	Machine learning/AI
CoDa	Cooperation Databank	https://cooperationdatabank.org/	Cooperation in social dilemmas
Hi-Knowledge	Hi-Knowledge	https://hi-knowledge.org/	Invasion Biology (Science and nature)

Figure 4.4: This is the resulting Dataframe after executing the snippet of code in the Listing 4.3 in the Jupyter Notebook.

4.4 ORKG Comparisons

As previously stated, comparisons are one of the entities that can be retrieved using the ORKG R package. However, unlike other entities, it was far more efficient to have it displayed directly as a matrix rather than in a list format, where it would be harder to work on, process, and compare with other data.

In order to retrieve a comparison, the ORKG connector will be utilized once again, and its attribute “comparisons” will be given the ID of the required comparison. As a result, a matrix is returned containing the corresponding ORKG comparison data (Figure 4.5).

The conversion of ORKG comparison data into a Dataframe reduces friction in using comparison data, since the data are readily available in a data structure that is native to the language of the computing environment, in our case R. Hence, the automated conversion increases the interoperability of data and eases data integration and processing.

```
In [22]: library(orkg)
orkg <- ORKG(host="https://orkg.org")
orkg$comparisons$get("R196200")[2:5,1:3]
```

	The Effects of Vocabulary Knowledge and Dictionary Use on EFL Reading Performance/Contribution 1	The effect of dictionary training in the teaching of English as a foreign language/Contribution 1	The Effect of Using Dictionary to Develop Students' Vocabulary in MTs. Al-Musthofa/Contribution 1
location	Hangzhou, China	Alcalá de Henares, Spain	Surabaya, Indonesia
method	quiz (reading comprehension and translation)	- quiz (translation, vocabulary, grammar, collocations and idioms); -training; -questionnaire	- observation; -vocabulary test
research problem	Correlation between scores on vocabulary size, specific vocabulary knowledge and reading comprehension	Possibility of improving students' knowledge about dictionary and their linguistic competence through a systematic dictionary training	- Dictionary users' enthusiasm when using dictionary; -Impact of a paper dictionary on vocabulary size growth
sample size	110 participants in total (96 valid samples, 14 invalid)	51 participants in total	56 participants in total (30 in experimental group; 26 in control group)

Figure 4.5: The matrix displayed after retrieving the comparison with the ID R196200.

4.5 Materializing templates

Template materialization is the approach that enables generating code dynamically based on the ORKG template specifications. This was solved using two approaches that are standard functions in R: “eval parse text paste”¹ and “function that instantiates instances in a list format”. The first allows the conversion of a string into executable code. Having that feature allows both naming variables with string values and using these variables by calling their name in string format. So that allows the use of the properties of a given template, which are in a string format. The second approach resulted from the inability to create new classes or alter existing ones due to a locked environment during the import of the ORKG package. This approach, put simply, is building a constructor for lists. This means having a function that initializes a list, which will play the role of the new class. The reason is that we can

¹<https://goodscienceblog.wordpress.com/2016/12/02/a-very-useful-function-eval-parse-text-paste/>

attribute to lists a new class that will refer to the type list. In addition, these lists can have indexes that can point to variables and functions. Having a constructor allows checking the type of arguments given by the specification of a template. With these two strategies in hand, the next moves are: First, we retrieve the template's name, properties, and types etc. from the template specification. Second, we determine if any of its components are themselves nested templates; if so, we repeat the previous step for each nested template until we locate one that is not nested. Using the first method, we begin constructing the function for this last by concatenating these pieces of data in a manner that will allow it to be executed later. This string is then executed with the approach "eval parse text paste" to define the constructor function. We key the preprocessed name of the templates in the hash list that indexes to its function definition. Preprocessed name refers to a name that has been modified so that it can be given to a variable. Finally, we repeat that process indefinitely until all of the templates have been visited. After completing all of the steps, the hash list will contain all of the functions needed to construct a template-based list for each of the templates acquired during this process.

To restate, "materialising the templates" in the R package refers to the process of creating a function with type-specific arguments that follows the template's specification and builds list instances, when populated with the given values.

Before populating the templates, the user should have the possibility of printing documentation in scripts about the template. Thus, each instance will be provided with its doc-string, which can be displayed by passing the value "doc" to the argument "text" of the function that creates this instance, which is stored in the `materialized_templates` of the attribute `templates` of the ORKG connector (the first block in the Figure 4.6).

4.6 Using templates and serializing data

After materialising the templates, printing their documentation, and populating them with the appropriate data, a template-based list is generated. The class of this list will be the preprocessed name of its template. This list will contain the filled properties of the templates, the host used to access these templates, and indexes pointing to two functions.

Pretty print, the first function, aids in printing this list as JSON data. And as indicated in the last chapter, the package includes a feature that allows data serialisation, which is the second function. It refers to the process of converting these template-based data structures into JSON-LD, a storable format.

Figure 4.6 illustrates the steps of template population and the template data serialization into JSON-LD (Listing 4.4). In the example, the “p-value” template was already selected and materialized, along with its nested-template “scalar value specification”. Then, a string and a numeric value were assigned to the template’s parameters, the value specification and the label, respectively. As a result, an instance, or the template-based data, was created, which was afterwards serialized into JSON-LD using the method `serialize_to_file`.

```
tp = orkg$templates$list_templates()
tp$pvalue(text= 'doc')
```

```
Creates a template of type R12006 (p-value)
  :param label: the label of the resource of type string
  :param value_specification: a nested template, use orkg.templates.scalar_value_specification
  :return: a string representing the resource ID of the newly created resource
```

```
tp = orkg$templates$list_templates()
instance = tp$pvalue(label= paste('the p-value', sep=''),
                    value_specification = tp$scalar_value_specification(
                      label = "0.000000131112475", 0.000000131112475)
                    )
```

```
instance$serialize_to_file('article.contribution.json', format='json-ld')
```

Figure 4.6: The documentation of the template and the serialization process.

Listing 4.4: The output of the serialization of an instance of the p-value template, the JSON-LD file.

```
{
  "@id": "_:n1",
  "label": "the p-value",
  "@type": [
    "https://orkg.org/class/C1003"
  ],
  "P9003": [
    {
      "@id": "_:n2",
```

```
"label": "0.0000000131112475",
"@type": [
  "https://orkg.org/class/C1004"
],
"P9004": [
  "1.31112475e-08"
]
}
],
"@context": {
  "label": "http://www.w3.org/2000/01/rdf-schema#label",
  "P9003": "https://orkg.org/property/P9003",
  "P9004": "https://orkg.org/property/P9004"
}
}
```

4.7 The conversion from R's Dataframe to annotated table

Metadata about a table may be stored independently from the table itself, expanding the use of an annotated model of tabular data². If an implementation is provided a file containing tabular data, this CSVW annotation helps machines parse and interpret its contents. An annotated table (Listing 4.5) is the fundamental annotated data type, with core annotations including an id and label, a type of `https://orkg.org/class/Table`, and annotated columns and rows. Each annotated column (Listing 4.6) in this table has a unique identifier, a descriptive name, and a numeric position relative to the other annotated columns in this table; these are all of the `https://orkg.org/class/Column` constant type. And then, there are annotated rows (Listing 4.7), which are defined by a title that correspond to the name of the row, if there is one; otherwise, it would be assigned using this format: “Row X”, where X represents its index; and annotated cells (Listing 4.8). These final attributes are characterized by a constant type `https://orkg.org/class/Cell`, a value and a column index.

²<https://www.w3.org/TR/2015/REC-tabular-data-model-20151217/>

Listing 4.5: Annotated table.

```
"P4015": [
  {
    "@id": "_:n4",
    "label": "To fill",
    "@type": [
      "https://orkg.org/class/Table"
    ],
    "columns": [...],
    "rows": [...]
  } ]
```

Listing 4.6: Annotated Column.

```
"columns": [
  {
    "@type": [
      "https://orkg.org/class/Column"
    ],
    "label": "non-failing heart (NF)",
    "name": "non-failing heart (NF)",
    "number": 1
  },...]

```

Listing 4.7: Annotated Row.

```
"rows": [
  {
    "@type": [
      "https://orkg.org/class/Row"
    ],
    "title": "Row 1",
    "label": "Row 1",
    "number": 1,
    "cells": []
  },...
]
```

Listing 4.8: Annotated Cell.

```
"cells": [  
  {  
    "@type": [  
      "https://orkg.org/class/Cell"  
    ],  
    "value": "99.0"  
  },...  
]
```

4.8 Tabular data display in ORKG

CSVW compliant data can be displayed as a table in ORKG. An additional feature of the ORKG interprets and displays this annotation alongside other resources. This annotation implies that its instance is of class ORKG: Table. In terms of structure, this instance is displayed similarly to other resources. However, it can be, in addition, visualized in a tabular chart (Figure 4.3) by clicking the table icon next to the resource's label (Figure 4.2).

Chapter 5

Applications

In this chapter, we demonstrate the presented ORKG R package in practice by means of two applications, one in life science and the other in soil science. The two applications are good examples that demonstrate the key features of the ORKG R package: ORKG template-based production of machine readable scientific information and use of ORKG data, in particular resources with ORKG:table type, in R scripts.

5.1 Life science application

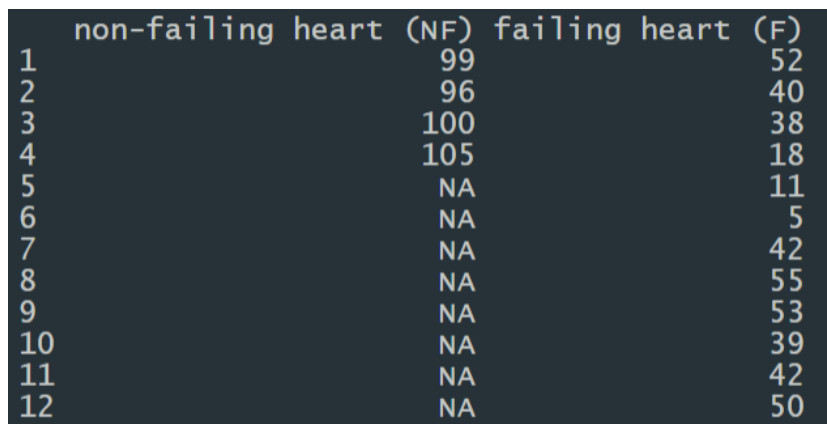
The life science use case is based on the paper by Haddad et al. titled “Iron-regulatory proteins secure iron availability in cardiomyocytes to prevent heart failure” [11]. Specifically, we focus on the result stating that “IRE binding activity was significantly reduced in failing hearts”.

We acquired some data, which was plotted in the article, in CSV format from the principal investigator of the original work (see Figure 5.1). The tabular data is for the two groups Non-Failing Hearts (NF) and Failing Hearts (F) and is the input data in the t-test with dependent variable IRE binding activity and with output p-value (approximated as $p < 0.001$ in Figure 1 B in the original work). Our goal is to represent information about the t-test conducted in the original work in machine readable form, using the ORKG R package and the ORKG Student’s t-test template¹. This template contains the following properties:

- `label`: A label for the t-test

¹<https://orkg.org/template/R12002>

- **has dependent variable:** The study design dependent variable, which in this case is iron-responsive element binding, and will be provided with a URI, which is considered an R Character Class.
- **has specified input:** An instance of ORKG:table, which will be assigned the tabular input
- **has specified output:** It has a template p-value, and the p-value is nested to the template value specification, which will then have the script's approximation of the p-value.



	non-failing heart (NF)	failing heart (F)
1	99	52
2	96	40
3	100	38
4	105	18
5	NA	11
6	NA	5
7	NA	42
8	NA	55
9	NA	53
10	NA	39
11	NA	42
12	NA	50

Figure 5.1: The targeted Dataframe from the research paper.

We developed an R script to replicate the t-test as shown in Listing 5.1.

Listing 5.1: Here we read the CSV file and use its content to compute the P-value.

```
df = read.csv("data.csv", check.names=FALSE)
tt = t.test(df[["non-failing heart (NF)"]],
            df[["failing heart (F)"]],
            var.equal=FALSE)
pvalue = tt$p.value
pvalue_ceil = ceiling(pvalue * 1000) / 1000.0
pvalue_str = format(pvalue, digits=9, big.mark = ",", scientific =
FALSE)
```

Given the computed p-value, we materialize the student's t-test (R12002) using the ORKG R package as shown in Listing 5.2. This automatically materialises all nested templates, specifically the p-value and value specification templates.

Listing 5.2: So here we set the host and materialize the template Student’s t-test and all its nested templates.

```
orkg <- ORKG(host="https://sandbox.orkg.org")
orkg$templates$materialize_template(template_id = "R12002")
```

Finally, we instantiate the template to produce a machine readable description of the t-test conducted in the original work. The values are assigned to the corresponding arguments of the template, as shown in Listing 5.3. Note that the last line serializes the machine readable description to a JSON-LD file name `article.contribution.1.json`.

Listing 5.3: The portion of code that illustrated how information is inserted into the template.

```
instance <- tp$students_ttest(
  label=paste("Statistically significant hypothesis test with IRE
    binding dependent variable on failing and non-failing hearts
    (p<\",pvalue_ceil,)\", sep=\"\"),
  has_dependent_variable="http://purl.obolibrary.org/obo/G0_0030350",
  has_specified_input=tuple(df, "Summary data showing
    iron-responsive element (IRE) binding activity in LV tissue
    samples"),
  has_specified_output=tp$pvalue(paste("the p-value of the
    statistical hypothesis test (p<\",pvalue_ceil,)\", sep=\"\"),
    tp$scalar_value_specification(pvalue_str, pvalue)
  ),
)
instance$serialize_to_file("article.contribution.1.json",
  format="json-ld")
```

The serialized data file can be submitted, together with the original manuscript, to a journal for review. Upon acceptance, the publisher can ensure that the article relates to machine readable scientific information as supplementary material in DOI metadata. This linking can be implemented by using the “IsSupplementedBy” relation available in DOI metadata. Listing 5.4 demonstrates the linking of a test DataCite DOI² for the article by Haddad et al. and the supplementary material in form of a JSON-LD file containing a machine readable description of the t-test conducted in the original work.

²<https://api.test.datacite.org/doi/10.7484/16y2-1t51>

Listing 5.4: Portion of the metadata of the DOI, where it shows the life science supplementary material.

```
"relatedIdentifiers": [
  {
    "relationType": "IsSupplementedBy",
    "relatedIdentifier":
      "https://zenodo.org/record/7337659/files/article.contribution.ls.json",
    "relatedIdentifierType": "URL"
  }[...]]
][...]
```

Given the article DOI, systems such as ORKG can now simply harvest the content. Specifically, in ORKG we can add the paper by DOI lookup (Figure 5.2). As a result, the metadata including the title of the paper and authors etc. are displayed. The machine readable contribution description (here for the t-test) is also harvested. After selecting the research field, users will see the contribution data automatically displayed (Figure 5.3).

General paper data

Paper DOI or BibTeX ?

<https://api.test.datacite.org/doi/10.7484/16y2-1t51>

Lookup result

Paper title: Iron-regulatory proteins secure iron availability in cardiomyocytes to prevent heart failure

Authors: Haddad, Saba, Wang, Yong, Galy, Bruno, Korf-Klingebiel, Mortimer, Hirsch, Valentin, Baru, Abdul M., Rostami, Stephanie, Renner, André, Toischer, Karl, Zimmermann, Fabian, Engeli, Stefan, Jordan, Jens, Bauersachs, Johann, Hentze,

Publication date:

Published in: Oxford University Press (OUP)

Figure 5.2: Metadata after looking up the DOI link.

As specified by the template for Student’s t-test, the three properties of the contribution “Statistically significant hypothesis test with IRE binding dependent variable on failing and non-failing hearts ($p < 0.001$)” are included in contribution data with corresponding values. Since ‘has specified input’ is of type table, ORKG

shows a table icon next to its label (see arrow in Figure 5.3). When this button is pressed and the table is shown (Figure 5.4), it is straightforward to see that it is identical to the original Dataframe used in the conducted t-test.

Iron-regulatory proteins secure iron availability in cardiomyocytes to prevent heart failure ★ 79

August 2016 40 citations Life Sciences Saba Haddad Yong Wang Bruno Galy Mortimer Korf-Klingebiel

Valentin Hirsch Abdul M. Baru Fatemeh Rostami Marc R. Reboll Jörg Heineke Ulrich Flögel Stephanie Groos

André Renner Karl Toischer Fabian Zimmermann Stefan Engeli Jens Jordan Johann Bauersachs Matthias W. Hentze

Kai C. Wollert Tibor Kempf

Published in: *European Heart Journal* DOI: <https://doi.org/10.1093/eurheartj/ehw333>

Statistically significant hypothesis test with IRE binding dependent variable on failing and non-failing ▼

Preferences

has specified input	https://github.com/markusstocker/doi-10-1093-eurheartj-ehw333/blob/main/data.csv Summary data showing iron-responsive element (IRE) binding activity in LV tissue samples 📄
has specified output	the p-value of the statistical hypothesis test ($p < 0.001$)
has dependent variable	http://purl.obolibrary.org/obo/GO_0030350

Figure 5.3: Properties filled automatically, after setting the research field and a directional arrow pointing to the visualization button.

And then, for instance, this Dataframe is to be directly retrieved from ORKG, with one of the results mentioned, which is converting tables into a R Dataframe. In order to illustrate how this can be useful for the aims of the data analysis, the resulting Dataframe is then plotted in a boxplot. These two steps are depicted in two figures: Figure 5.5 and Figure 5.6.

5.1. Life science application

View dataset: Summary data showing iron-responsive element (IRE) binding activity in LV tissue samples

non-failing heart (NF) ▾	failing heart (F)
Search 12 records...	Search 12 records...
96	40
99	52
100	38
105	18
nan	50
nan	42
nan	39
nan	53
nan	55
nan	42
nan	5
nan	11

« < 1 of 1 > »

Figure 5.4: The tabular visualization of the life science use case, after clicking pressing the visualization button.

```
df <- orkg$resources$by_id('R219781')$as_dataframe()
df$NF <- df[['non-failing heart (NF)']]
df$F <- df[['failing heart (F)']]
boxplot(df[3:4],
        data=df,
        cex.lab=0.65,
        xlab="Fig. 1 IRE binding activity for non-failing (NF) and failing (F) hearts.",
        ylab="IRE binding activity (%)",
        col="orange",
        border="brown",
        boxwex = 0.4,
        ylim = c(0, 120)
)
```

Figure 5.5: Here is a sample of code that displays the table in a boxplot after retrieving it from ORKG.

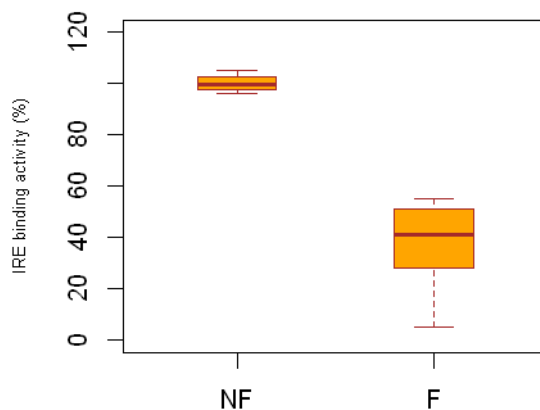


Fig. 1 IRE binding activity for non-failing (NF) and failing (F) hearts.

Figure 5.6: The boxplot represents a summary of the data shown in the ORKG table.

5.2 Soil science application

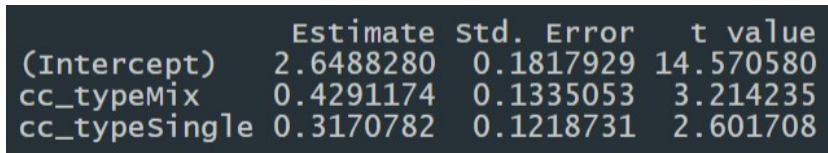
The soil science use case will go through the same process, but in this instance two contributions will be produced. As a result, the article DOI will relate to two JSON-LD files as supplementary material. The data utilised in this use case is a subset of the data provided by the researcher, whose work will shortly be published. These two Dataframes will be utilised to construct the two contributions and are displayed in the two figures (Figure 5.7 and Figure 5.8).

	Estimate	Std. Error	t value
(Intercept)	2.6488280	0.1831154	14.465350
cc_variantMustard	0.3264725	0.1577686	2.069312
cc_variantClover	0.3434370	0.1577686	2.176840
cc_variantOat	0.2752331	0.1577686	1.744536
cc_variantPhacelia	0.3231701	0.1577686	2.048380
cc_variantMix4	0.3438099	0.1577686	2.179203
cc_variantMix12	0.5144248	0.1577686	3.260628

Figure 5.7: The Dataframe used for the first contribution.

Both contributions make use of the model fitting template³, whose materialization is represented in the Listing 5.5. This template relies on three other nested templates

³<https://orkg.org/template/R222407>



	Estimate	Std. Error	t value
(Intercept)	2.6488280	0.1817929	14.570580
cc_typeMix	0.4291174	0.1335053	3.214235
cc_typesingle	0.3170782	0.1218731	2.601708

Figure 5.8: The Dataframe used for the second contribution.

to complete its structure: the statistical model template⁴, the formula template⁵, and the value specification template⁶. Each doc-string of these templates are displayed in the Figure 5.9

Listing 5.5: This listing shows the population of the template and its serialization for one contribution.

```
orkg <- ORKG(host="https://sandbox.orkg.org/")
orkg$templates$materialize_template(template_id = "R222407")
tp = orkg$templates$list_templates()
```

```
Creates a template of type R222437 (Formula)
: param label: the label of the resource of type string
: param has_value_specification: a nested template, use orkg.templates.value_specification
: return: a string representing the resource ID of the newly created resource

Creates a template of type R222407 (Model Fitting)
: param label: the label of the resource of type string
: param has_output_dataset : a parameter of type Table (which is here considered tuple )
: param has_input_model: a nested template, use orkg.templates.statistical_model
: param has_input_dataset : a parameter of type URI (which is here considered character )
: return: a string representing the resource ID of the newly created resource

Creates a template of type R222432 (Statistical Model)
: param label: the label of the resource of type string
: param is_denoted_by: a nested template, use orkg.templates.formula
: return: a string representing the resource ID of the newly created resource

Creates a template of type R222439 (Value Specification)
: param label: the label of the resource of type string
: param has_specified_value : a parameter of type String (which is here considered character )
: return: a string representing the resource ID of the newly created resource
```

Figure 5.9: The doc-string of the templates used in the soil science use case.

⁴<https://orkg.org/template/R222432>

⁵<https://orkg.org/template/R222437>

⁶<https://orkg.org/template/R222439>

Therefore, in order to follow the example more in-depth, below is a snippet of code that illustrates the first contribution that will assist in making the template's structure more clear.

Nota bene: Since this work has not yet been published and we are primarily interested in the Dataframe, just the data extraction will be discarded in that snippet of code. In addition to that, `df1` represents the first Dataframe extracted from the data provided in the data analysis phase.

Listing 5.6: This listing shows the population of the template and its serialization for one contribution.

```
instance <- tp$model_fitting(
  label="Linear mixed model fitting with MWD as response, CC
    variant as predictor variable, and soil depth as random
    variable",
  has_input_dataset=
  "https://github.com/markusstocker/gentsch22cover/blob/main/df.MWD.csv",
  has_input_model=tp$statistical_model(
    label="A linear mixed model with MWD as response and CC
      variant as predictor variable",
    is_denoted_by=tp$formula(
      label="The formula of the linear mixed model with MWD as
        response and CC variant as predictor variable",
      has_value_specification=tp$value_specification(
        label="MWD_cor ~ cc_variant + (1|depth)",
        has_specified_value="MWD_cor ~ cc_variant + (1|depth)"
      )
    )
  ),
  has_output_dataset=tuple(df1, "Results of LMM with MWD as response
    and CC variant as predictor variable")
)
```

After populating the template for each contribution and serializing their instances as shown in the Listing 5.6 (which was done for only one contribution), two JSON-LD files will be generated: `article.contribution.1.json` and `article.contribution.2.json`,

which will be referred by the test DataCite DOI⁷, as shown in the following Listing 5.7.

Listing 5.7: The presentation of the two supplementary materials in DOI.

```
relatedIdentifiers": [  
  {  
    "schemeUri": null,  
    "schemeType": null,  
    "relationType": "IsSupplementedBy",  
    "relatedIdentifier":  
      "https://zenodo.org/record/7366513/files/article.contribution.1.json",  
    "resourceTypeGeneral": null,  
    "relatedIdentifierType": "URL",  
    "relatedMetadataScheme": null  
  },  
  {  
    "schemeUri": null,  
    "schemeType": null,  
    "relationType": "IsSupplementedBy",  
    "relatedIdentifier":  
      "https://zenodo.org/record/7366513/files/article.contribution.2.json",  
    "resourceTypeGeneral": null,  
    "relatedIdentifierType": "URL",  
    "relatedMetadataScheme": null  
  }  
]
```

The final step is to add the two contributions to ORKG alongside the metadata of the paper, after which it's all out of hand because ORKG will do the job of discovering both the metadata and these contributions via the given DOI, which is exactly what happened, as shown in the Figure 5.10. The properties are filled exactly as implied by the instances produced in the computational environment, and the Dataframe with the type of table can be visualized as indicated in the Figure 5.11)

Then, this ORKG table resource displayed in Figure 5.11 will be analysed by retrieving it directly from ORKG and displaying it as three bar plots (Figure 5.13). Figure 5.12 illustrates the corresponding code.

⁷<https://api.test.datacite.org/doi/10.7484/s06c-8y98>

Cover crops improve soil structure and change OC distribution in aggregate fractions

[Ancient History \(Greek and Roman through Late Antiquity\)](#)
[Gentsch, Norman](#)
[Laura Riechers, Florin](#)
[Boy, Jens](#)


[Schweneker, Dörte](#)
[Feuerstein, Ulf](#)
[Heuermann, Diana](#)
[Guggenberger, Georg](#)

Published in: *SOIL*

ked model fitting with MWD as... **Linear mixed model fitting with MWD as...**

Preferences

has input model [A linear mixed model with MWD as response and CC variant as predictor variable](#)

has output dataset [Results of LMM with MWD as response and CC variant as predictor variable](#)


has input dataset <https://github.com/markusstocker/gentsch22cover/blob/main/df.MWD.csv>

Figure 5.10: The exhibition of the soil science use case and the button that lead to the tabular visualization.

View Tabular Data: Results of LMM with MWD as response and CC type as predictor variable ×

Options ⋮

	Estimate	Std. Error	t value
	<input type="text" value="Search 3 records..."/>	<input type="text" value="Search 3 records..."/>	<input type="text" value="Search 3 records..."/>
(Intercept)	2.648828	0.1817929	14.57058
cc_typeMix	0.4291174	0.1335053	3.214235
cc_typeSingle	0.3170782	0.1218731	2.601708

« < **1 of 1** > »
Go to page:
Show 10 ▾

Figure 5.11: The tabular visualization of the second Dataframe presented in the soil science use case in ORKG.

```
df <- orkg$resources$by_id('R253297')$as_dataframe()
for (i in 1:length(st)){
  df1 <- st[,i, drop=FALSE]
  barplot(t(as.matrix(df1)),beside=TRUE, names =rownames(st),
          ylab=colnames(df1),cex.axis=1.5, cex.names=0.9)
}
```

Figure 5.12: In this snippet of code, the ORKG table was retrieved and then plotted in three bar graphs.

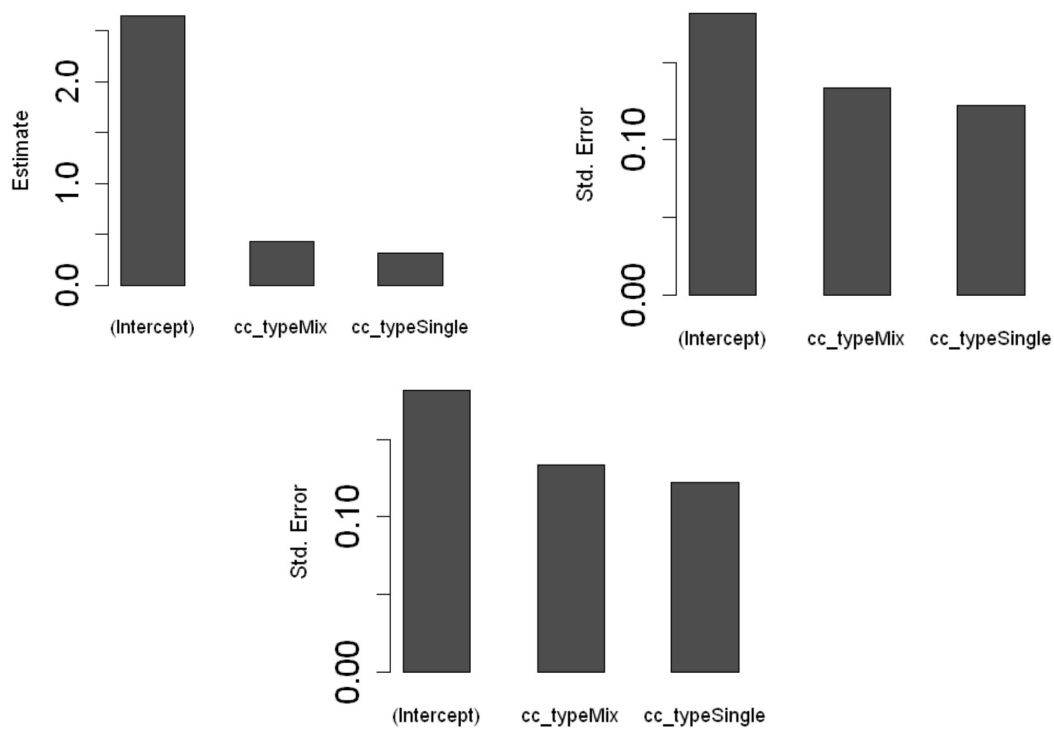


Figure 5.13: The source of the three bar graphs was the table retrieved from ORKG.

Chapter 6

Discussion

We first revisit the research questions and discuss how we have addressed them with the proposed solutions. We then present the advantages of the solutions in contrast to the presented related works. Finally, we discuss the limitations and highlight potential further work.

6.1 Discussion of research questions

RQ1 aims to ensure that scientific information is produced machine readable. the “pre-publication production of the scientific information” is the answer to this first question. This was demonstrated by the aforementioned two use cases, in which the supplementary information was supplied with the manuscript and injected into the DOI metadata, where it was subsequently discovered and harvested automatically by ORKG.

RQ2 aims to dynamically generate programmatic interfaces in R based on ORKG templates. This question was resolved by obtaining all the information about the ORKG template from the API and using it to construct a function that complies with the template definition and, when filled with data, generates lists. These instances of type list represent the template based data.

RQ3 aims to ensure that machine readable scientific information is published and can be harvested, given article DOIs. The publication of this scientific knowledge is ensured by submitting the produced supplementary material together with the manuscript to a journal for review. Alternatively, researchers can deposit this supplementary material in a research data repository of their choice. The supplementary material in our case was the JSON-LD file, which was the output of the

serialization of the template-based data. As for the Harvesting of this information, it is enabled by linking supplementary material in the DOI metadata of the article.

6.2 Advantages of the proposed approach

Having this scientific data produced in a machine-readable form directly from the content producer is the key difference compared to NLP- or crowdsourcing-based post-publication information extraction approaches. Because they are inaccurate (the performance of automated approaches is generally low), result in shallow information (coarse rather than fine granular information), and are hard to scale (crowdsourcing relies on massive collaboration).

The fact that the machine-readable information produced during the data analysis phase does not go directly to ORKG is also significant because it implies that researchers submit supplementary material to publishers and published alongside the original work. Due to the availability of these links and the machine-readable metadata, any system can use this to harvest the content and exploit it to generate value, which is inexpensive.

The R package provides additional useful features. With automatic conversion of ORKG tabular data resources and comparisons data into data frames we ensure least friction in reusing ORKG content for the widest possible types of use. This is useful not only for publication-related matters, but also for any user who wants to work with this data, visualise it, or compare it to other data.

Finally, creating code dynamically based on the most current version of ORKG template specifications is a significant feature, as it scales well and saves a great deal of effort in the event that a researcher wishes to modify a template's specification or add and use new ones. As soon as the template's specification is adjusted, it can be used without having to deploy or update the package.

6.3 Limitations of the work

One of the limitation of the presented approach is the challenge in ensuring that researchers submit structured, machine-readable data alongside their research papers; in other words, adopt the ORKG R package in data analysis. Considerable awareness and capacity needs to be built to ensure broad adoption.

Additionally, the serialization of given data into JSON-LD, a feature provided by both the R package and the Python package, is still restricted to certain data types, such as strings and numeric values, that are easily identifiable and, at the

highest level, data frames that adhere to the CSVW data model so that they can be interpreted as a table and visualized as a table by ORKG. However, all other sorts of data types are considered as strings by default.

The R package, unlike the Python package, has not yet been published in CRAN, therefore it cannot be imported directly. Instead, the user must construct a binary package from the source code, which is not intuitive, and then install and import this binary package locally. Not to mention that this package still has missing features, as it focuses on only retrieving and not altering or creating directly entities into ORKG from a given computational environment.

6.4 Future work

As indicated earlier, the serialization is still restrictive, supporting only a few data types or restricting each research contribution to a single template(nested template). Instead of having anything serialized interpreted as a research contribution, the output of a template serialization could be just a part of a research contribution. Therefore, a further improvement is to enhance the capacity of serialization by expanding its content in terms of the various types of data that it feeds into an annotated model on one side. And on the flip side, the capacity of a single research contribution can be enhanced in terms of the number of serializations it can acquire. This can be achieved by combining template-based data within a single research contribution. The package can also be maintained to have more interactions with the ORKG API than those currently supplied, such as sending POST requests to create and alter resources directly in ORKG. As for encouraging researchers to use the proposed approach, one possibility is to make it first a standard norm in the institute where ORKG originated to serve as an example for other academic institutions and thereby encourage them to follow suit.

Chapter 7

Conclusion

This thesis developed an ORKG R package and demonstrated its use in data science, in particular to efficiently reuse ORKG content in data analysis and to ensure scientific information produced in data analysis is produced machine readable. This approach stands in stark contrast with post-publication information extraction approaches, and has several advantages, in particular the production of rich machine readable scientific information (fine granular), the unmatched accuracy, the technological simplicity (especially in contrast to NLP-based approaches), and user friendliness (since no manual data entering in ORKG is needed).

We have presented work related to this thesis along three subgroups: scholarly information infrastructure (Section 2.1), data repositories (Section 2.2), and methods (Section 2.3), including manual and automated extraction. Each was briefly compared to ORKG, which served as the benchmark platform. Before presenting the architecture, a more thorough design containing ORKG's environment for a tangible perspective (Section 3.2), a generic strategy was offered to cope with this latter (Section 3.1). Afterward, the findings that exhibit approaches used in this architecture were displayed (Chapter 4), including, for example, ORKG comparison retrieval, dynamic materialization of templates, and serialization of the instance produced by the filling of these templates, all of which were addressed by the R package implemented during this thesis. We have demonstrated the proposed approach and tools in two applications in different disciplines: life science and soil science (Chapter 5). As the applications clearly show, the proposed solution successfully contributes to the research questions driving this work. The applications underscore the effectiveness of the technique in terms of time savings, reusability, and interoperability. Despite the significant advantages that come with this approach, it must be integrated into research workflows in order for its utility to be readily apparent. This can be ac-

complished by building capacity among researchers as well as increasing the number of systems that make effective use of the published machine readable scientific information (Chapter 6).

Bibliography

- [1] Donat Agosti and Willi Egloff. “Taxonomic information exchange and copyright: the Plazi approach”. In: *BMC Research Notes* 2.1 (Mar. 2009), p. 53. ISSN: 1756-0500. DOI: 10.1186/1756-0500-2-53. URL: <https://doi.org/10.1186/1756-0500-2-53>.
- [2] Rubayyi Alghamdi and Khalid Alfalqi. “A Survey of Topic Modeling in Text Mining”. In: *International Journal of Advanced Computer Science and Applications* 6 (Jan. 2015). DOI: 10.14569/IJACSA.2015.060121.
- [3] Claudio Atzori et al. “The OpenAIRE Workflows for Data Management”. In: Oct. 2017, pp. 95–107. ISBN: 978-3-319-68129-0. DOI: 10.1007/978-3-319-68130-6_8.
- [4] Sören Auer et al. In: *Bibliothek Forschung und Praxis* 44.3 (2020), pp. 516–529. DOI: doi: 10.1515/bfp-2020-2042. URL: <https://doi.org/10.1515/bfp-2020-2042>.
- [5] Andreas Blumauer. “From Taxonomies over Ontologies to Knowledge Graphs”. In: *Semantic Web Company* (Aug. 2016). URL: <https://blog.semanticweb.at/2014/07/15/fromtaxonomies-over-ontologiesto-knowledge-graphs>.
- [6] Helena Cousijn et al. “Connected Research: The Potential of the PID Graph”. In: *Patterns* 2 (Jan. 2021), p. 100180. DOI: 10.1016/j.patter.2020.100180.
- [7] Kamel Fadel. “Data Science with Scholarly Knowledge Graphs”. MA thesis. 2021. DOI: 10.15488/11535.
- [8] Martin Fenner. “Tracking the Growth of the PID Graph”. en. In: (2019). DOI: 10.5438/BV9Z-DC66. URL: <https://blog.datacite.org/tracking-the-growth-of-the-pid-graph/>.
- [9] Martin Fenner and Amir Aryani. “Introducing the PID Graph”. In: (2019). DOI: 10.5438/JWVF-8A66. URL: <https://blog.datacite.org/introducing-the-pid-graph/>.
- [10] Georgios Gkatzelis et al. “The global impacts of COVID-19 lockdowns on urban air pollution”. In: *Elementa: Science of the Anthropocene* 9 (Apr. 2021). DOI: 10.1525/elementa.2021.00176.
- [11] Saba Haddad et al. “Iron-regulatory proteins secure iron availability in cardiomyocytes to prevent heart failure”. In: *European Heart Journal* 38 (Aug. 2016), ehw333. DOI: 10.1093/eurheartj/ehw333.
- [12] Robert Huber et al. “Integrating data and analysis technologies within leading environmental research infrastructures: Challenges and approaches”. In: *Ecological Informatics* 61 (2021), p. 101245. ISSN: 1574-9541. DOI: <https://doi.org/10.1016/j.ecoinf.2021.101245>. URL: <https://www.sciencedirect.com/science/article/pii/S1574954121000364>.

-
- [13] Mohamad Yaser Jaradeh et al. “Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge”. In: *Proceedings of the 10th International Conference on Knowledge Capture*. K-CAP ’19. Marina Del Rey, CA, USA: Association for Computing Machinery, 2019, pp. 243–246. ISBN: 9781450370080. DOI: 10.1145/3360901.3364435. URL: <https://doi.org/10.1145/3360901.3364435>.
- [14] Farzad Kiani and Oguzhan Tas. “A survey Automatic Text Summarization”. In: vol. 5. June 2017, pp. 205–213. DOI: 10.17261/Pressacademia.2017.591.
- [15] Markus Krötzsch and Gerhard Weikum. “Journal of Web Semantics: Special Issue on Knowledge Graphs”. In: *Semantic Web* (Aug. 2016). URL: <http://www.websemanticsjournal.org/index.php/ps/announcement/view/19>.
- [16] Jing Li et al. *A Survey on Deep Learning for Named Entity Recognition*. 2018. DOI: 10.48550/ARXIV.1812.09449. URL: <https://arxiv.org/abs/1812.09449>.
- [17] Tianjing Li, Julian PT Higgins, and Jonathan J Deeks. “Collecting data”. In: *Cochrane Handbook for Systematic Reviews of Interventions*. John Wiley Sons, Ltd, 2019. Chap. 5, pp. 109–141. ISBN: 9781119536604. DOI: <https://doi.org/10.1002/9781119536604.ch5>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119536604.ch5>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119536604.ch5>.
- [18] C. Lichten et al. *Citizen Science: Crowdsourcing for Research*. THIS.Institute, 2018. ISBN: 9781999653903. URL: <https://books.google.de/books?id=y4G6zQEACAAJ>.
- [19] Paolo Manghi et al. “OpenAIRE Research Graph Dump”. In: (Dec. 2019). DOI: 10.5281/zenodo.3516918.
- [20] Barend Mons. “Which gene did you mean?” In: *BMC Bioinformatics* 6.1 (June 2005), p. 142. ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-142. URL: <https://doi.org/10.1186/1471-2105-6-142>.
- [21] Allard Oelen, Markus Stocker, and Sören Auer. “TinyGenius: Intertwining Natural Language Processing with Microtask Crowdsourcing for Scholarly Knowledge Graph Creation”. In: *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*. JCDL ’22. Cologne, Germany: Association for Computing Machinery, 2022. ISBN: 9781450393454. DOI: 10.1145/3529372.3533285. URL: <https://doi.org/10.1145/3529372.3533285>.
- [22] Heiko Paulheim. “Knowledge graph refinement: A survey of approaches and evaluation methods”. In: *Semantic Web* 8 (Dec. 2016), pp. 489–508. DOI: 10.3233/SW-160218.
- [23] Jason Priem, Heather Piwowar, and Richard Orr. “OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts”. In: (May 2022).
- [24] Jay Pujara et al. “Knowledge Graph Identification”. In: vol. 8218. Oct. 2013, pp. 542–557. DOI: 10.1007/978-3-642-41335-3_34.
- [25] *re3data.org: NEON*. <http://doi.org/10.17616/R36G6R>. Accessed: 2022-11-23.
- [26] *re3data.org: PANGAEA*. <http://doi.org/10.17616/R3XS37>. Accessed: 2022-11-23.
- [27] Wei Shen, Jianyong Wang, and Jiawei Han. “Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions”. In: *Knowledge and Data Engineering, IEEE Transactions on* 27 (Feb. 2015), pp. 443–460. DOI: 10.1109/TKDE.2014.2327028.

- [28] *SN SciGraph FAQ*. <https://scigraph.springernature.com/explorer/faq/>. Accessed: 2022-11-19.
- [29] Giuliana Spadaro et al. “The Cooperation Databank: Machine-Readable Science Accelerates Research Synthesis”. In: (Oct. 2020). DOI: 10.31234/osf.io/rveh3.
- [30] Mark D. Wilkinson et al. “The FAIR Guiding Principles for scientific data management and stewardship”. In: *Scientific Data* 3.1 (Mar. 2016), p. 160018. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18. URL: <https://doi.org/10.1038/sdata.2016.18>.
- [31] Yuhong Yuan and Richard Hunt. “Systematic Reviews: The Good, the Bad, and the Ugly”. In: *The American journal of gastroenterology* 104 (June 2009), pp. 1086–92. DOI: 10.1038/ajg.2009.118.