

GOTTFRIED WILHELM LEIBNIZ UNIVERSITÄT HANNOVER
FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIK

Publishing machine actionable reproducible scholarly knowledge

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

BY

Anouar Ganfoud

Matriculation number: 10000755

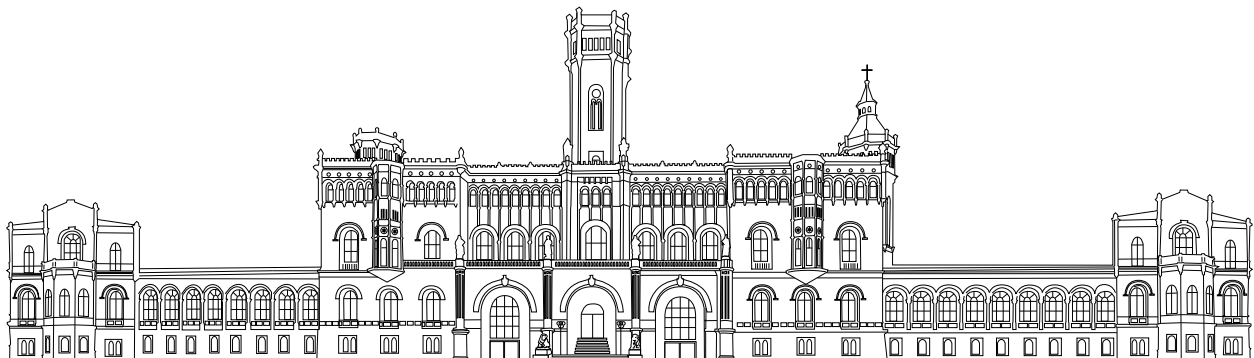
E-mail: anouar.ganfoud@stud.uni-hannover.de

First evaluator: Prof. Dr. Sören Auer

Second evaluator: Dr. Markus Stocker

Supervisor: Dr. Markus Stocker

09 Mai 2021



Declaration of Authorship

I, Anouar Ganfoud, declare that this thesis titled, 'Publishing machine actionable reproducible scholarly knowledge' 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.

Anouar Ganfoud



Signature:

Date: 09.05.2021

Acknowledgements

First and foremost I am extremely grateful to my supervisor, Dr. Markus Stocker for his invaluable advice, continuous support, and patience during my master thesis. His immense knowledge and plentiful experience have encouraged me in all the time of my academic project and daily life. I would also like to thank the ORKG team and especially Mohamad Yaser Jaradeh and Kheir Eddine Farfar for their technical support on my study. I would like to thank all the members of Stencila, in particular Nokome Bentley and Alexander Ketchakmadze. I also appreciate all the support I received from Jose Carvalho and Marc Bria from the OJS community. It is their kind help and support that have made my study and life a wonderful time. Finally, I would like to express my gratitude to my parents, my two brothers, my friends and my workmates. Without their tremendous understanding and encouragement in the past few months, it would be impossible for me to complete my study.

Abstract

Scientific research faces many challenges related to the credibility of published results. In essence, there is typically not enough documentation on how experiments are conducted and data is generated. Thus, increasing the reliability of articles through reproducibility will improve the quality of the published scientific literature and offers better reliable results. This thesis describes today's problem of the research literature related to non-reproducibility and unstructured data such as weak experiments designs, errors, data dredging and under-specified methods. We suggest a variety of solutions to resolve these problems through linking machine readability with the reproducibility of the information in academic papers. We use therefore a knowledge platform which provides reproducibility on one side and on the other side another platform that ensures the machine actionability of data. Then, we build an integration between them and test it on a selected use case article. After establishing the integration, we obtained, as a result, a reproducible article described in machine-actionable and structured manner. Thereafter, we created a solution that allow every reader to switch between the static and dynamic (reproducible and machine-readable) form of the article.

This thesis discusses the benefits and limitations of these observed results and emphasizes the future alternatives.

Keywords: Reproducibility, Reproducible Science, Machine actionability, Reproducible scholarly knowledge

Contents

1	Introduction	1
2	Background	5
2.1	Reproducible research	5
2.2	FAIR Data	6
2.2.1	Findable	6
2.2.2	Accessible	7
2.2.3	Interoperable	7
2.2.4	Reusable	7
2.3	Machine-actionable data	7
2.4	Scholarly knowledge	9
2.5	RM-ODP architecture	10
2.5.1	Enterprise viewpoint	11
2.5.2	Information Viewpoint	11
2.5.3	Computational Viewpoint	11
2.5.4	Engineering Viewpoint	12
2.5.5	Technology Viewpoint	12
3	Related Work	13
3.1	CERN Analysis Preservation	13
3.1.1	Overview	13
3.1.2	Concept	13
3.1.3	Technology	14
3.1.4	UI and functionality	15
3.2	Whole Tale Research Environment	17
3.2.1	Overview	17
3.2.2	Concept	18
3.2.3	Technology	18

3.2.4	UI and Functionality	21
3.3	Integrating eLife Magazine with Stencila	21
3.3.1	Overview	21
3.3.2	Concept	22
3.3.3	Technology	22
3.3.4	UI and Functionality	23
3.4	D4Science: an e-Infrastructure for Virtual Research Environments	25
3.4.1	Overview	25
3.4.2	Concept	25
3.4.3	Technology	26
3.4.4	UI and Functionality	27
4	Approach	31
4.1	Overview of the system	31
4.1.1	Description of the used platforms	32
4.1.2	Description of the system from the ODP viewpoints	48
4.2	Description of the proposed solution	52
4.2.1	Creation of executable articles	52
4.2.2	Integration	53
5	Application	56
5.1	Converting the paper to a reproducible form	56
5.1.1	Reproducibility with Stencila	56
5.1.2	Reproducibility with Jupyter Book	62
5.2	Integration	68
5.2.1	Integration between ORKG and Stencila	68
5.2.2	Integration between OJS and Stencila	76
6	Discussion	86
6.1	Advantages	86
6.2	Limitations and challenges	87
6.3	Comparisons	88
6.3.1	Comparing Stencila with Jupyter Book	88
6.3.2	Comparison with related work	90
6.4	Future works	92
7	Conclusions	94
	Bibliography	96

List of Figures

3.1	The architecture of the CERN Analysis Preservation platform [44].	14
3.2	First part of data submission in CAP [44].	15
3.3	Second part of data submission in CAP [44].	16
3.4	Permissions setting for an analysis record on CAP [44].	17
3.5	Architecture of Tale [53].	19
3.6	Whole Tale API [53].	20
3.7	Whole Tale Dashboard UI [53].	21
3.8	Visualization of plots and graphics in Stencila [66].	23
3.9	UI of the eLife static version [68].	24
3.10	UI of the eLife dynamic version [68].	24
3.11	The D4Science Workspace platform [79].	27
3.12	The D4Science social networking platform [79].	28
3.13	The D4Science data analytics platform [79].	29
3.14	The D4Science data publishing platform [79].	29
4.1	The main components of the system and their interactions	31
4.2	Stencila personal page of the user.	34
4.3	Stencila files management page.	35
4.4	Stencila Snapshot Button.	35
4.5	ORKG Interface of the research contribution.	38
4.6	ORKG Interface for comparison between contribution.	39
4.7	ORKG Interface of a Dataset template.	40
4.8	Visualization of a paper in the form of a graph in ORKG.	40
4.9	OJS Reader Interface [95].	43
4.10	Dashboard interface in OJS 3.x [95].	44
4.11	Generating plot with code chunk in Jupyter Book [97].	46
4.12	Adding the “hide-content” tag to a cell in Jupyter Lab.	46
4.13	Adding interactivity through Binder, ThebeLab and Google Colab.	47
4.14	The Stencila-ORKG-OJS system of our study	48
5.1	The creation of a new project interface in Stencila.	57

5.2	The Gentsch paper output in Stencila.	60
5.3	Output of the plot after running it in Stencila.	61
5.4	Output of the p-values table after running it in Stencila.	62
5.5	Gentsch’s paper markdown output in Anaconda Jupyter Notebook GUI . .	63
5.6	Output of p-values table generated from the computation of code’s cell. . .	64
5.7	HTML files in ”_build” folder generated by Jupyter Book.	66
5.8	Output of the Gentsch paper with Jupyter Book.	67
5.9	Computation of the Gentsch paper on Binder.	67
5.10	Output of the Gentsch plot with Jupyter Book.	68
5.11	ORKG package in PyCharm.	70
5.12	Gentsch’s paper presentation in ORKG.	74
5.13	Visualizing the dataset in the form of a tabular.	75
5.14	Maximum significant p-value output in ORKG.	75
5.15	Issue data in OJS.	78
5.16	Creating a submission in OJS: Policy section.	79
5.17	Creating a submission in OJS: Defining the metadata.	80
5.18	Adding a galley to a submission in OJS.	81
5.19	Index of the Gentsch journal in OJS.	82
5.20	Output of the Gentsch article in OJS.	83
5.21	Output of the computed version of Stencila in OJS.	84
5.22	Output of the machine-actionable version of ORKG in OJS.	85

List of Tables

4.1	Stencila plugin list [81].	33
6.1	Stencila Hub VS Jupyter Book.	89

Acronyms

API Application Programming Interface

CAP CERN Analysis Preservation

CSV Comma Separated Values

DOI Digital Object Identifier

FAIR Findable Accessible Interoperable Reusable

GUI Graphical User Interface

HTML HyperText Markup Language

JATS Journal Article Tag Suite

JSON JavaScript Object Notation

ODP Open Distributed Processing

OJS Open Journal Systems

ORCID Open Researcher Community ID

ORKG Open Research Knowledge Graph

VRE Virtual Research Environment

XML eXtensible Markup Language

YAML YAMl Ain't Markup Language

Chapter 1

Introduction

Several researches concerning the reliability and efficiency of scientific research are improved every day. Indeed, the innovation and prosperity known in computer science have a remarkable impact on scientific publications. This progress of scientific research is often summed up as, “*If I have seen further, it is by standing on the shoulders of giants*” (This quote has been attributed to Sir Isaac Newton since it appeared in a letter he wrote to fellow English scientist Robert Hooke [1]). However, statistics demonstrate that the majority of research is not reproducible [2].

Scientists became worried by problems in some sensible experiments such as drug development while using results of studies realized in other companies. They have only the possibility to replicate the underpinning research, which represents about one quarter of studies [3]. Similarly, the researchers of the German company “*Amgen*” faces many problems with analyzing foreign data and approve it only in 11% of “landmark” studies [4].

Thus, scientists cannot assume the credibility of the data based on the information included in published articles. He needs to know how the author gathered it. Otherwise, the consequences could be probably grim if an important amount of science is not reproducible. At the outset, it is also necessary to admit that not all scientific research are pretended or granted to develop in findings which are related to applications for human health. Outstanding research and great ideas may also lead to a dead-end. Researchers have to accept that because it defines simply the nature of research. Efficiency of 100% and waste of 0% is unlikely to be obtainable. According to an internet-based survey of 1,576 researchers in *Nature*, about 90% of respondents confirm that there is either a slight or significant crisis of reproducibility in scientific research [5]. However, considerable improvement can be achieved by making research reproducible.

Moreover, the replicability of scientific experiments plays a significant role in removing the complexity of different systems. Reproducibility will offer the possibility for the scientist to replicate the results of a previous research based on the same data as were exploited by the original author. That is, a second scientist might need the same dataset to set up the same analysis files and develop the same statistical approaches in order to obtain the same results. Reproducibility can be defined as a minimum fundamental quality for a finding to be credible and informative [6]. An independent scientist should be capable to replicate an experiment for a research project. Besides, the same outcome should be achieved under the same circumstances. It is also important to be able to recreate the same conditions. Nevertheless, it is hard to guarantee the same circumstances in practice, even in tightly controlled experiments. “Experiment” should be understood in a large sense, including also computational work. Hence, there is an increasing importance in assuring and evaluating the reproducibility and clearness of the published scientific literature.

In addition to reproducibility issues, published scholarly articles remain nowadays mere analogues of their print relatives [7]. It becomes crucial to automatically process scholarly knowledge which is communicated in the form of digital scholarly documents [8]. In fact, the communicated information is usually ambiguous and hard to reproduce [9]. However, scientists and researchers point to reproduce academic knowledge in digital as it used to be in print [10]. The discussion of all these challenges is clearly debated in many articles related to the interdisciplinary research [11]. The inter-linking of article contents is not machine actionable and difficult to implement. Consequently, the automatic processing of scholarly knowledge still not yet achieved by any publication.

Furthermore, traditional document-based scholarly communication still facing the challenge of the digital transformation seen in the last few years in other information rich publishing and communication services. In fact, many publications for several services such as street maps, phone books or encyclopedia are not just digitized but also cannot show the implementation of entirely new means of information management and access. As far as we know, these services are distinguished from scholarly communication and considered as completely novel approaches to information management, sharing, access, collaborative reproducing and processing.

Additionally, there are further problems related to scholarly knowledge publications. In fact, recent scientific discourse is hard to follow, due to exponential enlargement of scientific knowledge volume [12, 13]. Nowadays, the research paper publishing is witnessing a major reform, and individual scientists cannot manage the huge amount of publications, which can certainly proceed to an unclear, ambiguous and redundant research publications. Indeed, the structuring and organization of

the data will be more difficult when the volume of publications doesn't stop growing. Therefore, the volume issue can be as well considered as one of the reasons causing the reproducibility crisis.

It becomes necessary to establish a flexible, concrete, context-sensitive and machine actionable representation of scholarly knowledge and its related framework for knowledge processing, sharing and publishing. Indeed, this representation should be structured, interconnected and semantically full. Knowledge graphs, for instance, can be used as an appropriate approach for representing ambiguous information in an organized manner. Although the big importance of the technology responsible for the establishment of this representation, many other concepts are as well required such as the way how the scholarly knowledge will be gathered and generated during the entire research life cycle. Thus, it becomes required, on that side, to publish scientific knowledge in a flexible, one-grained, semantic, and context-sensitive way, and on the other side, to include a reproducibility approach to this representation.

In this thesis, we will suggest a solution to improve the reproducibility of machine actionable scholarly knowledge. We devise an overall framework, implement the framework and test it by building on existing systems, e.g. Stencila [14] as a platform for authoring and executability of the science, the Open Research Knowledge Graph (ORKG) [15] as an infrastructure for the collecting, curation, processing and publication of machine-actionable scholarly knowledge, and Open Journal Systems as a framework for the submission and publication of the academic literature. Hence, we integrate the two aspects of machine actionability and reproducibility through an integration of two platforms, ORKG and Stencila, respectively. We will also prototype the impact of reproducible science on traditional online publication platforms by integrating Stencila with the Open Journal Systems (OJS) [16]. We will present, evaluate and discuss all these integrations and their benefits for scholarly communication.

In this work, we tackle the following research questions:

1. How can we ensure that scholarly knowledge is reproduced and published both in human and machine-actionable form ?
2. How to describe a scientific article in a structured human-machine-actionable manner ?
3. How can we demonstrate that the human and machine-readable versions of the knowledge, generated in research work, can be published ?

The remainder of this thesis is structured as follows. In Chapter 2, we describe some basic knowledge related to our thesis. All related works are presented in the

Chapter 3. The details of the proposed approach are highlighted in Chapter 4 which describes as well the components of the system used for implementing the given solution. Chapter 5 presents the results of the implementation with discussing the details of our proposed technical solution. The advantages and drawbacks of the used approach, as well as the evaluation of frameworks and the future works are presented in Chapter 6. We close this thesis with summarizing the ideas and the obtained results in Chapter 7.

Chapter 2

Background

This chapter introduces the main concepts underlying this thesis, specifically reproducible research, FAIR data and machine-actionability of data, scholarly knowledge and RM-ODP architecture.

2.1 Reproducible research

Reproducible research can be defined as the availability of research data and code so that readers are able to replicate the results as claimed in scientific articles.

The concept of reproducibility is basically applied to the scientific approach, the pillar of Science, and especially during the formulation of a hypothesis, the collection and analysis of the data and also while developing, executing and reporting a study. The main goal of reproducible research is typically understood as ensuring the corroboration and verification of data correctness. However, reproducible research aims to improve transparency and credibility in the computational sciences. Computer scientist Jon Claerbout defined reproducibility as a software framework and set of procedures that enable the reader of a paper to discover the whole processing path from the raw data and code to tables and plots [17]. Several scientific domains take advantage of this approach and use it intensively in their data researches such as clinical trials [18], computational biology [19], finance and economy [20], and epidemiology [21]. Furthermore, the National Science Foundation (NSF) [22] defines reproducibility as the ability of research to duplicate the outcome of an earlier study basing on the same materials as were used by the original author. Thus, the documentation of reproducibility imposes the sharing of analytical datasets, statistical code, appropriate metadata and relevant software. Reproducible science aims to avoid issues with trust for the representation of data and analyzes. However, the

definition does not indicate to what degree deviations are tolerable. According to the NSF, reproducibility does not provide a new evidential weight that is more highly trusted. Therefore, NSF adds a new evidence called “replicability” and defined as “the ability of a researcher to duplicate the results of a prior study if the same procedures are followed, but new data are collected” [23]. With this definition, it remains unclear which operational criteria can create a successful replication or reproduction. In addition, this terminology is not universally adopted and can sometimes be wrongly explained. Besides, other issues could be identified which mainly concern the interpretation, reporting, design, analysis and corroborating studies. In fact, a study about the use of this terminology in the scientific literature demonstrates that there are many other similar alternative and intermingling of concepts. For instance, the replication of some experiments in psychology reported in a study titled “Estimating the reproducibility of psychological science” [24] definitely relate the term “reproducibility” to the new studies conduct.

2.2 FAIR Data

The “FAIR Guiding Principles for scientific data management and stewardship” [25] were published in 2016 in “Scientific Data” to specify the main standards for scientific data management. The authors aim to help the science community by providing some guidelines to enhance the Findability, Accessibility, Interoperability, and Reuse of (digital) data. Indeed, humans are nowadays obliged to support more and more computational systems due to the increase of the complexity, quantity and speed of data. The principles defined by the guideline reinforce machine readability represented in the capability of computational framework to find, access interoperate, and reuse data without the need of any human intervention.

2.2.1 Findable

It represents the first principle described in the guideline. It consists in finding the data in order to reuse it. In fact, the data and metadata should be easy to find by both machines and humans. Besides, the FAIRification process requires some necessary factors such as the machine-readable metadata which is fundamental for automatic detection of datasets and services. The findable principle ensures that the humans and computer systems should not take too much time to find the needed metadata and data. Standard machine-actionable descriptive metadata enables the detection of interesting services and datasets.

2.2.2 Accessible

After finding the needed data, the second step is to determine the approach of their accessibility which include the authentication and authorization principles. According to this principle, the data and metadata should be stored for the long term in a way that machines and humans will be able to access, locally use and download them without any difficulty thanks to the standard communication protocols.

2.2.3 Interoperable

In common cases, the data should be integrated with other data in a way that permit the users able to interoperate it with different applications or workflows for analysis, storage and treatment. Besides, this principle ensures the exchange, the interpretation and the combination of the data in a (semi)automated manner with other data sets by humans as well as computer systems.

2.2.4 Reusable

FAIR aims mainly to improve the reuse of data. Thus, the description of metadata and data should be perfectly achieved to ensure their replication and combination in different settings. Indeed, data and metadata are adequately good described to enable data to be reused in future studies, offering the possibility of integration with diverse compatible data sources. Suitable citation have to be simplified, and the circumstances of the data adaptation should be understandable for machines and humans.

2.3 Machine-actionable data

Machine-actionable data, or computer-readable knowledge represents the data in a format in order to be structured and processed by a computer. The US Open Government Act describes machine-actionable data as "data in a format that can be easily processed by a computer without human intervention while ensuring no semantic meaning is lost" [26]. According to this definition, non-digital literature such as hand-written or printed articles can be classified as not machine-actionable due to their non-digital type. Besides, PDF files always enclose data which can be represented in the form of tables. The data in these tables are absolutely digital but not computer-readable due to the complexity of their access. However, a human will not find any difficulty to read it. Thus, to make this tabular data readable by the

computer, an author should write it in a structured format such as spreadsheet or CSV. Another example is the photos or scans of a paper which contain a non machine-actionable data, although the same data was represented, before the printing, in a text format such as a simple ASCII text file which can be machine-actionable and treatable. There are two groups of machine-actionable data:

- Marked up human-readable data that is also readable by machines. Examples of this format include HTML, RDFa and microformats.
- Machine-readable data in formats that are mainly designed for machine processing, e.g. XML, CSV, JSON and RDF.

Machine-actionability can be defined as data digitally reachable. A digitally accessible article is a document which is online published in order to allow humans to easily open it via computers. However, this document is not easy to extract, convert and manipulate via simple machine programming logic if it is not computer-readable. Machine-actionable approaches represent the cornerstone to recreate computational environments, or to provide the hole actual computational environment that the analysis was conducted in [27, 28]. For example, Dockerfiles and Vagrantfiles are used as machine-readable plain text approaches for directing virtual machines to a suitable specification [29].

A prediction, for instance, is considered machine-actionable when the user can automatically decide if the prediction is confirmed by the data or not. Even though computational reproducibility is nowadays becoming more and more popular as user-friendly tools are regularly being developed, there are no current solutions that convert, e.g., hypothesis tests descriptions into machine-actionable and reusable data. Researchers will be able to access to this data to load all the information related to analytical predictions. For instance, the metadata file should be enough to calculate or reach without difficulty effect volumes from performed analytical tests when a fully reproducible process is exploited and data can be reached as a piece of meta-data file [30].

During the peer study process, an automatic evaluation of computer-actionable hypotheses has at least two suitable procedures. Firstly, a forecast for the future shows that scientists will be enforced to submit completely computationally reproducible analysis scripts including their submissions. The computational reproducibility of the reviewed results in a manuscript must be inspected by editorial deputy or reviewers. Machine-actionable hypothesis tests would make this inspection an issue of executing a single function. Using R packages and machine-actionable format such as JSON or XML permit the creation of scripts that can be automatically executed in other languages.

However, reproducible science requires the employment of machine-actionable techniques such as JSON descriptor. Basically, data packages are based on this descriptor file. These descriptors could be easily treated by several tools during data analysis. They support metadata for the accumulation of resources, and a design for a tabular data. The machine-actionable adopts accessible and distributed language, and has provision to include descriptions, and information concerning subscribers and sources for each resource, which allows the connection to other existing metadata and assure data provenance. Hence, a machine-readable file can be extensible and can be extended to hold supplementary information as required.

Entirely new search retrieval assistance techniques could be reached thanks to machine readability. Using Alexa or Google Now to ask about research problems in an interactive way may be in the future possible [31].

2.4 Scholarly knowledge

A scholarly knowledge or academic discipline is a subdivision of knowledge that taught and studied at the college or university level [32]. This knowledge is created and published by the academic journals that are related to scientific research, and the scholarly societies and scientific departments or universities. Scholarly knowledge can be divided into various disciplines such as the scientific studies including mathematics, physics, chemistry, and earth science, and the humanities disciplines such as philosophy, religion, art and cultural learning; and the social science knowledge including sociology, history, economy, etc.

Peoples associated with scholarly knowledge are usually referred to as academics or experts. They are just people whose task it is to try to discover the best and most credible approaches of knowledge. Nowadays, everyone is able to create and publish the knowledge on the internet without facing any problem of claim and credibility. However, the art of detecting the best manners of knowing and separating fact and fiction is essential for all the readers, not just academics.

In several scholarly knowledge disciplines, the perfect and most creditable approach of knowing commonly includes the adaptation of different scientific means in some form. The application of these means is principally true for the natural sciences such as mathematics, physics, biology, etc. Nevertheless, a lot of social sciences and humanities like sociology, psychology, management and economics face many problems to apply scientific methods in one manner or another.

Other scholarly knowledge disciplines like mathematics and philosophy require a higher level of determinable reasoning as the main method of knowing. This may include the determination of what can be rationally deduced from what it is

already known or expected. Indeed, it becomes easier to determine what the first fundamental principles and the way of reasoning, especially with the complexities and deficiency of language. However, in many other disciplines of humanities such as art, history and literature, the level of reasoning and emphasis is very low, and doesn't require the application of scientific method. In fact, the used way of knowing is based principally on understanding the past, the prejudices, and the structure of the society and its parts.

In order to verify their knowledge and ensure high quality standards, academics usually adopt the peer review, which allows them to publish their knowledge only if it is classified credible by their peers. These peers are commonly two or three peer reviewers which are generally neutral and anonymous to the author. For example, Google Scholar [33] and the Web of Knowledge [34] are famous websites which contain publications of articles in peer-reviewed journals. These articles are frequently written in an academic language using technical wording, which make it hard to read for most of the peoples, so they rely on news outlets and popular research literature to convert these studies and translate it into simple understandable common language. However, this translation doesn't always lead the readers to the correct meaning and knowledge found by academics or researchers. Therefore, sometimes "scholarly knowledge" is a term that push us to refer to the original scholarly publication to get to the source of the material.

2.5 RM-ODP architecture

The Reference Model for Open Distributed Processing [35] is a standard invented jointly by the International Organisation for Standardisation (ISO) [36] and the International Telecommunications Union (ITU) [37]. Thanks to the exponential progress of the computer networking, computer systems around the world become highly interconnected. However, the inter-working between the systems remains poor due to the heterogeneity in interaction models. Open distributed processing (ODP) describes systems that ensure heterogeneous distributed processing both between and within systems via the adaptation of a standard interaction model. The aim of this reference model (RM-ODP) is providing a coordinating framework for the standardization of open distributed processing (ODP) through the creation of an architecture which supports distribution, inter-working, interoperability and portability.

RM-ODP contains four basic elements:

- An object modelling method to system requirements
- The requirement of a system based on interconnected viewpoints specifications

- The description of a system infrastructure using distribution transparencies for system adaptation
- A framework for evaluating system conformance

RM-ODP consists as well of four standards:

- Overview: contains scoping, justification and explanation of key concepts.
- Foundation: contains the definition of concepts and scientific framework for standardized description of distributed processing systems.
- Architecture: contains the specification of the necessary features that certify distributed processing as open.
- Architectural semantics: contains a definition of the modeling concepts.

RM-ODP describes a framework based on five generic and complementary viewpoints from which to abstract or observe ODP systems. Each viewpoint is composed of a list of structures, rules and concepts.

2.5.1 Enterprise viewpoint

The enterprise viewpoint [38] describes the business model and the business requirements. It focuses generally on the purpose, scope and policies. It covers the role of the systems in the business as well as the human user roles and business policies. We can describe this viewpoint using many key concepts such as purpose and targets, domain, activity, community, actors, role, scope, contact and policy.

2.5.2 Information Viewpoint

The information viewpoint [39] is employed to describe the semantics of information and the information processing. Various key concepts are adopted to ensure the information viewpoint like the information objects, association, contract, and policy.

2.5.3 Computational Viewpoint

The computational viewpoint [40] is used to specify the functionality of our system based on the interaction between the components and services that compose this system. These components are basically described through their interfaces. Indeed, the set of these interfaces displayed by a user, and the hidden set of other services

running in the system back-end represent the computational specification of our system.

Several interfaces and services are used to build the computational model such as the interface designed for the data processing, the Application Programming Interfaces (API) [41], or the services responsible for the actions like publish and subscribe or request and reply.

The computational viewpoint is composed of various key concepts including components, interactions, interface, binding, and quality of service (QoS).

2.5.4 Engineering Viewpoint

The engineering viewpoint [42] is used to define the design of distributed objects of our ODP system. It describes the infrastructure of the distributed systems. The engineering viewpoint is not based on the semantics of the ODP system, except to set up its specification for distribution and distribution transparency. The RM-ODP engineering viewpoint describes the structure of an ODP system. Several key concepts describe the engineering viewpoint such as cluster, capsule, nucleus object and node.

2.5.5 Technology Viewpoint

The technology viewpoint [43] focuses on the choice of technology, best practices and the implementation of the system. It can be also considered as the provision of a fundamental infrastructure.

Nevertheless, the selection of the technology may have several consequences. The provision of a particular quality of service could be guaranteed through the good choice of the technology, and based on the technology viewpoint feedback to other features of the system architecture. Moreover, the performance costs of interactions can be fixed thanks to the choices in the technology viewpoint. Thus, the quality of service which can be performed by the behaviour in other viewpoints can be as well determined through the choice in the technology viewpoint.

The technology viewpoint plays also an important role in the conformance testing development. It provides the information required for the interpretation of the observations used by a tester for the identification of the vocabulary and concepts adopted in the other viewpoints of the system requirements. For instance, it enables the recognition of valid interactions, in order to allow their appropriateness to be verified against some particular object behaviour.

Chapter 3

Related Work

A number of research projects have approached the idea of reproducibility in scientific literature. Most of them focus on presenting a solution for preserving and sharing their research objects (such as data, code, documentation, notes) throughout their research process to improve the reliability and credibility of their results.

3.1 CERN Analysis Preservation

3.1.1 Overview

CERN Analysis Preservation (CAP) [44] is a digital library service in High-Energy Physics [45] (HEP) based on a particular disciplinary research workflow. Indeed, this methodology aims to collect the research data analysis workflow steps and proceeding numerical objects. The importance of this approach can be represented by the use of the contextual knowledge needed to reproduce an analysis. Thus, CAP is defined as a mandatory phase towards better reuse of unique research literature and as a measure to simplify future reproducibility of results.

3.1.2 Concept

CAP implements a centralized framework which allows researchers to write directly their analysis with the launching of a new project. This can be considered an innovative form of reproducible research. It differs also from traditional approaches which usually separate between the documentation and preservation only after the analysis has occurred. Further, scientists are able to save the record of any feature or step of an analysis as well as relevant research aspects within their cooperation.

In fact, code, datasets, transitional documentation of processing steps, content and annotations or test processes could be submitted through the tool by scientists. CAP provides auto-completion of many elements of the analysis metadata based on the connection to the databases of the collaborations. Scientists can keep their implicit materials and share their data in a simple way. They will be able to also access research literature for future application. Besides, a capture of additional documents and analysis of the reusable data can be saved.

3.1.3 Technology

CAP is based on many recent technologies. The digital library platform Invenio [46] represents one of these technologies that permit CAP to create a custom digital repository solution adapted to fulfill different use cases, such as digital document repository, multimedia archive and integrated library structure. Indeed, Invenio supports storing JSON in its own digital repository database, then submits data to an Elasticsearch cluster that guarantees indexing and information retrieval. In addition to Invenio, JSON is also used as a solution to model the managed data in the form of a JSON Schema in order to assure the compliance of captured JSON snippets with the regular metadata specifications. A durable preservation of the captured properties can be guaranteed thanks to the Open Archival Information System (OAIS) [47] framework.

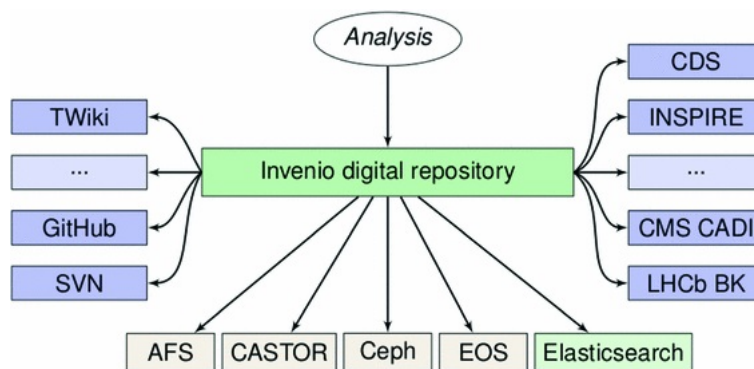


Figure 3.1: The architecture of the CERN Analysis Preservation platform [44].

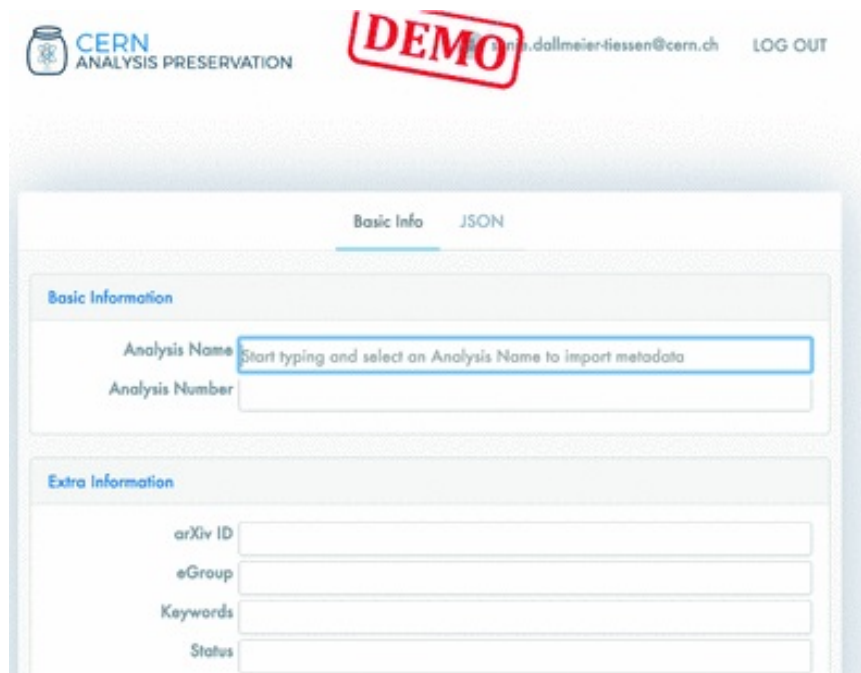
The figure above describes the mechanism of the CAP platform which includes :

- The "Kernel" of the Invenio framework.

- *EOS* [48]: Storage backend for running the file storage abstraction layer.
- *CASTOR* [49]: Connector aims to harvest finished datasets from different storage systems.
- *Git*: permit the connectors to harvest code used to build the analysis code.
- *TWiki* [50]: Connectors point to establish the interaction with the documentation systems.
- *CMS CADI* [51]: Internal collaboration frameworks used for ingesting the information to CAP.
- *CDS* [52], *INSPIRE*: Platforms where CAP can publish some parts of its confirmed open data.

3.1.4 UI and functionality

The UI supplies different features and entry points to serve various users, as somehow CAP will more easily become an essential factor of the research process.



The screenshot displays the CERN Analysis Preservation (CAP) web interface. At the top left is the CERN logo and the text "CERN ANALYSIS PRESERVATION". To the right is a red "DEMO" stamp, the email address "cristian.dallmeiertiessen@cern.ch", and a "LOG OUT" link. Below the header, there are two tabs: "Basic Info" (selected) and "JSON". The main content area is divided into two sections: "Basic Information" and "Extra Information". The "Basic Information" section contains two input fields: "Analysis Name" with a placeholder text "Start typing and select an Analysis Name to import metadata" and "Analysis Number". The "Extra Information" section contains four input fields: "arXiv ID", "eGroup", "Keywords", and "Status".

Figure 3.2: First part of data submission in CAP [44].

Figure 3.2 shows the submission form which allows the researchers to submit their content. This form represents the first part of submission that can be filled out automatically by using data from an existing database.

The screenshot shows a web form titled "DST selection". It is organized into several sections:

- Select a stripping line**: This section is the header for the form.
- Code**: Contains three fields: "LHCb code" with a green "+ Add New Item" button, "Platform" with a text input field, and "User code" with a green "+ Add New Item" button.
- Input Data**: Contains two fields: "Data" with a green "+ Add New Item" button and "MC Data" with a green "+ Add New Item" button.
- Output Data**: Contains two text input fields: "Data" and "MC Data".
- Stripping Line**: A single text input field.
- Trigger**: A single text input field.

Figure 3.3: Second part of data submission in CAP [44].

Figure 3.3 highlights the second part of the submission form which permits researchers to input their code and data. After submitting their information, researchers will be able to share their analysis internally with other collaborators. Figure 3.4 shows the permission interface which organizes the rights of each invitee. This feature has been considered very essential by the community because it allows them to follow the work progress of the CAP content.

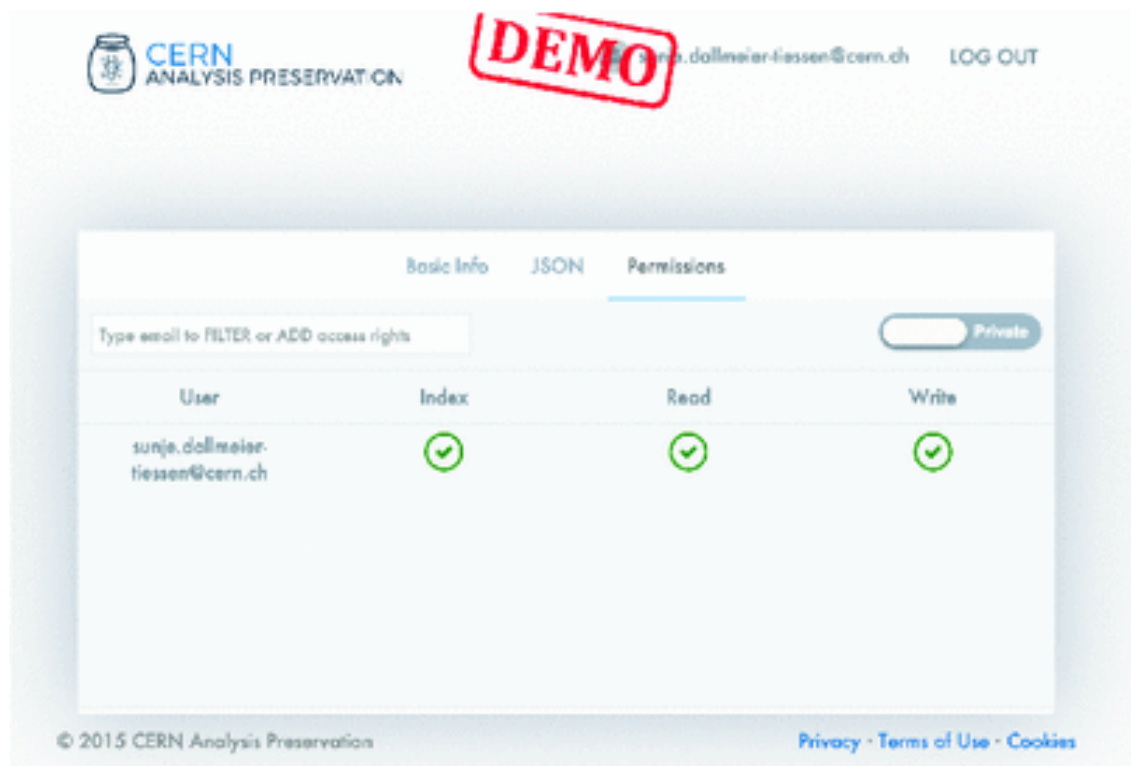


Figure 3.4: Permissions setting for an analysis record on CAP [44].

3.2 Whole Tale Research Environment

3.2.1 Overview

Whole Tale [53] is an open source, web-based, multi-user framework aims to simplify reproducibility for scientists. It allows them to build, publish and execute their tales or their executable research aspects through capturing code, data and the whole software environment aimed to create research findings. The Whole Tale project has been created through the community literature essentially based on groups input and collaborations with researchers. Indeed, the Whole Tale platform is used in various projects to train researchers for improving reproducibility as well as dealing with the Whole Tale framework in the classroom.

3.2.2 Concept

Whole Tale aims to allow researchers to determine and build computational environments in a simple way. They will be able to also control the entire conduct of computational experiments and publish them for analysis and reproducibility [54]. Moreover, the Whole Tale platform is being developed to facilitate the use of procedures that make reproducibility of computational research better understandable. The framework has two main goals:

- increase the transparency tolerance in a way that researchers can run each type of computational experiments.
- improve the infrastructure of computational experiment so that scientists will be more transparent.

The Whole Tale project maintains computational reproducibility by allowing scientists to write and package code, dataset and any type of information about the process and computational environment. Besides, this project supports the review of computational analysis results stated in published research. Whole Tale helps the developers and operators of research data repositories to face the challenge of fulfilling the requirements of their communities through offering a compatibility for new types of scholarly objects, approaches of access, and workflows for review and verification. It implements also reproducible definition by allowing specific citation of externally referenced data, recording the artifacts and provenance data required to simplify comprehension, transparency, and execution of the computational workflows and processes adopted for examination and reproducibility at the time of publication.

3.2.3 Technology

Architecture

Whole Tale is based on Tale [53] which is defined as an executable research object that fuses code (computational methods), data (references), computational environment, and narrative (old science story). Tales have a standards-based format complete with metadata. The Whole Tale framework enables users to interactively create and modify Tales and to re-execute a Tale to reproduce and investigate results as received by the original Tale creator.

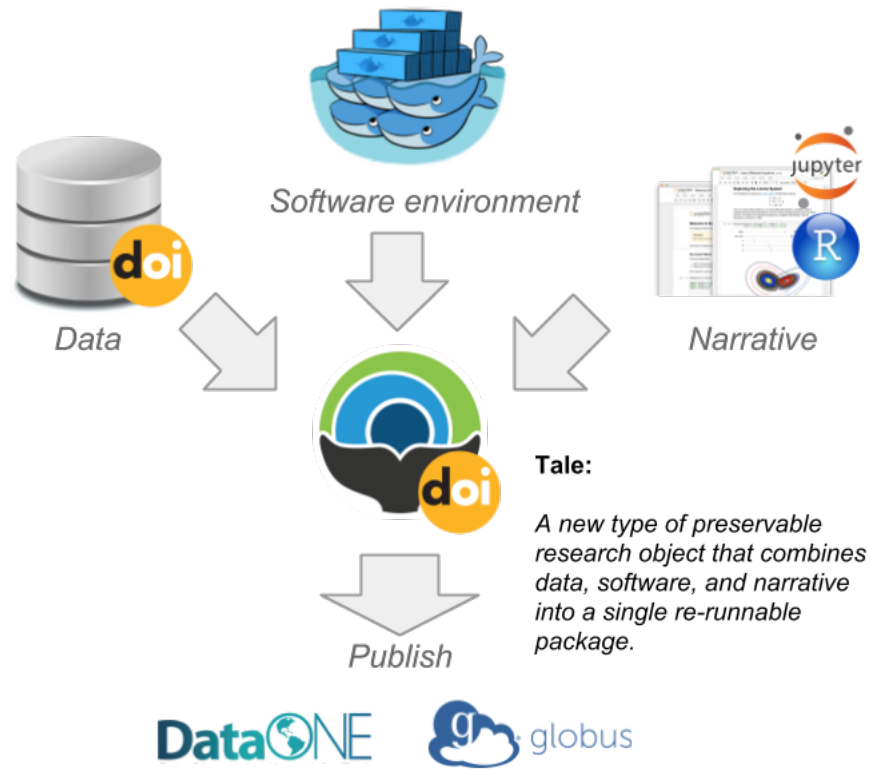


Figure 3.5: Architecture of Tale [53].

Figure 3.5 explains the functionality of Whole Tale which is based on:

- The data received from the contributor (DOI URL)
- A software environment to run the code and data such as Docker [55].
- Narrative which represents the code usually written in R or Python.

After receiving all the main elements needed for the process of computing, Whole Tale will generate the data and publish it on one of the cloud environment such as DataONE [56] or Globus [57].

Whole Tale API

The Whole Tale API enhances the Girder platform [58] by integrating Whole Tale features such as:

- Tales, Instances and Images
- Downloading data from remote repositories
- Shared home and Tale work-space repositories
- Pushing Tales to remote repositories
- Access and caching of remote data

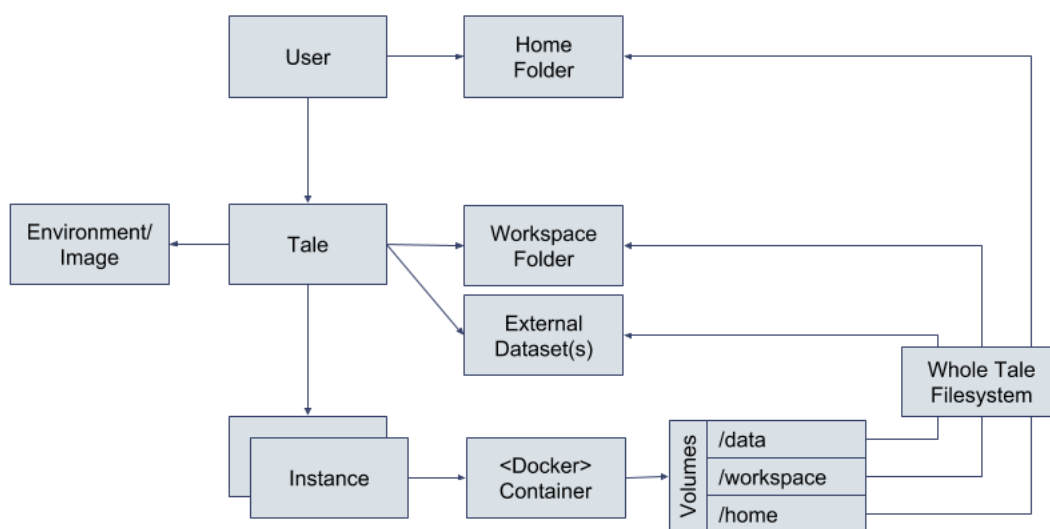


Figure 3.6: Whole Tale API [53].

The Whole Tale API supplies through Celery/Redis [59, 60] database an extensible framework:

- Creating and administrating Tale images
- Starting Tale instances such as RStudio [61], Jupyter Notebook [62]
- Generating data from external source

Figure 3.6 highlights the functionality of the Whole Tale API based on the interaction between the users and the Tale instances. In fact, RStudio or Jupyter define every Tale. Narrative, data, code are saved in work-space repositories. Basically, a Tale instance in the form of a running docker container will be mounted when the user runs a Tale.

3.2.4 UI and Functionality

Most of the features offered by Whole Tale such as starting, creating or sharing Tales are accessible through the dashboard interface. Indeed, the Whole Tale API is based on this interface, created through the Ember JavaScript open-source web framework [63].

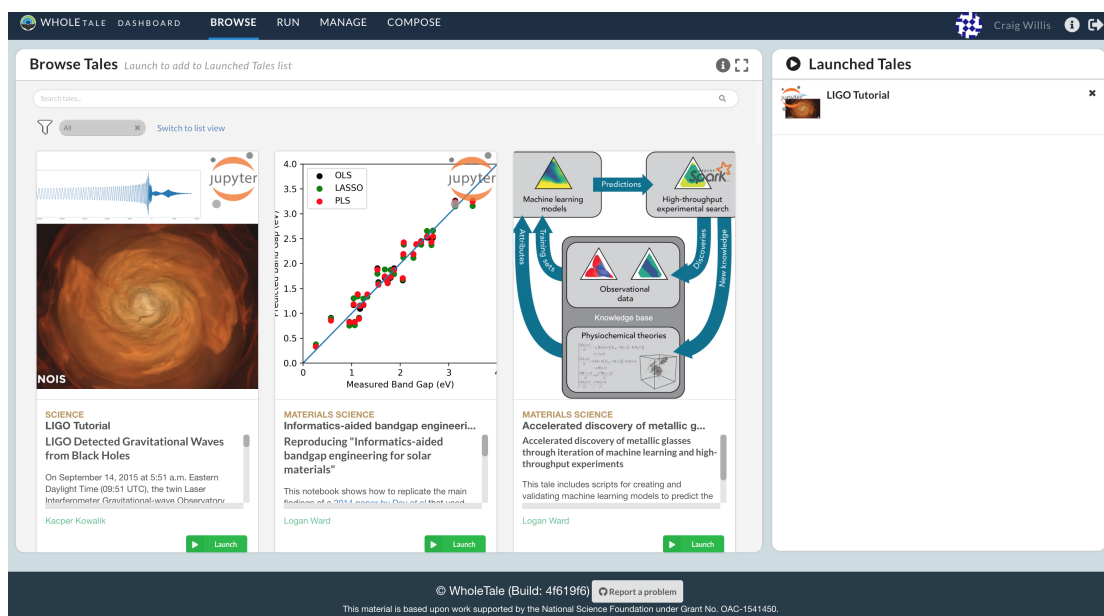


Figure 3.7: Whole Tale Dashboard UI [53].

Figure 3.7 describes the user interface of the Whole Tale dashboard. The user can browse tales and choose and run the corresponded one depending on the requirements. Furthermore, a user can access to the code, data and literature of each tale. A list of launched tales will be shown with the possibility of stopping and deleting them.

3.3 Integrating eLife Magazine with Stencila

3.3.1 Overview

Journals have different publishing specifications. A set of standards is outlined by stakeholders from academia and industry in order to define that research data should

be Findable, Accessible, Interoperable, and Reusable (FAIR) [25]. In 2018, eLife Magazine [64] launched its first reproducibility prototype by publishing a demonstration of a dynamic and code-based reproducible peer-reviewed article, based on the Stencila framework [14] and Binder [65]. This methodology allows data and analysis to be completely reproducible by the reader and cut with the old static representation of results based on traditional formats such as HTML or PDF.

3.3.2 Concept

The eLife-Stencila project points to be part of the larger vision to improve research literature, all the way from writing through to publication as a reproducible, self-contained document. This project facilitates the tasks of authors by bringing computationally reproducible research papers to more authors and publishers.

Journal-ready scientific manuscript's conception

For integrating executable article in eLife, Stencila's article editor builds on Texture [66], an open source editor that aims to visually manipulate JATS XML [67] documents (a standard largely adopted by researchers in their scientific journals). Indeed, the framework enables the user to concentrate on the research without taking care of layout information, which is handled during publishing. Stencila permits the extension of Texture with code cells during the process of the implementation of all the used elements such as figures, references and citations in order to generate computed, data-driven figures.

Source data and analysis spreadsheets conception

The eLife-Stencila project gives a big importance to datasets as an integral part of the publication. They are represented as independents spreadsheet documents containing structured data. Furthermore, Analysis and plots can be driven through the referenced data from research article. Excel sheets, for instance, can hold some mathematical formulas and function calls to execute computations directly in a spreadsheet.

3.3.3 Technology

A modern web technology is used to implement the Stencila user interface and running it in the browser to avoid any problem of operating system's compatibility. Thus, JavaScript will be used by the predefined functions accessible in Stencila to

launch the execution of the platform. These features could be run directly in the editor such as the *plotly()* function which permits the generation of effective, interactive visualizations thorough the Plotly’s JavaScript library.

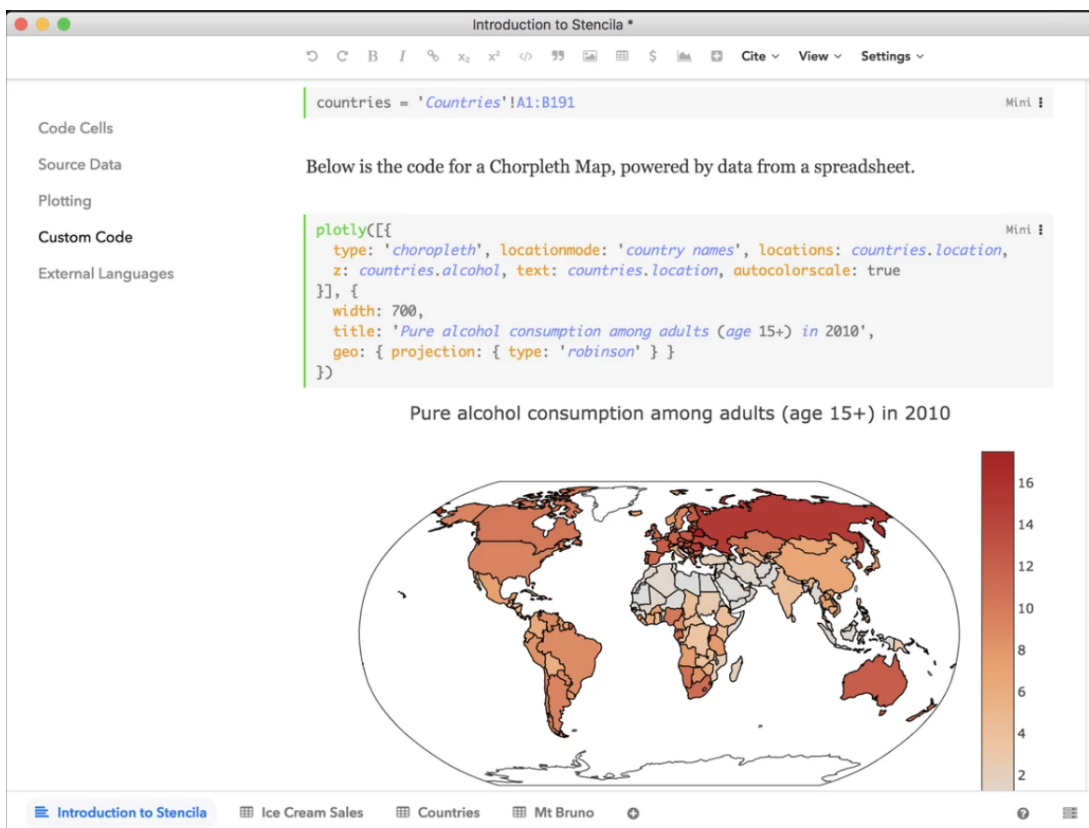


Figure 3.8: Visualization of plots and graphics in Stencila [66].

The integration between Stencila and eLife requires basically a connection to R, Python and SQL sessions in order to provide more advanced data analysis and visualization capacities. A track of dependency between code cells and spreadsheet (such as programming experience both in Stencila Sheets and Articles) will be conserved thanks to Stencila execution engine.

3.3.4 UI and Functionality

eLife magazine uses the HTML element *iframe* to integrate Stencila engine execution and permit users to jump between the static and dynamic version.

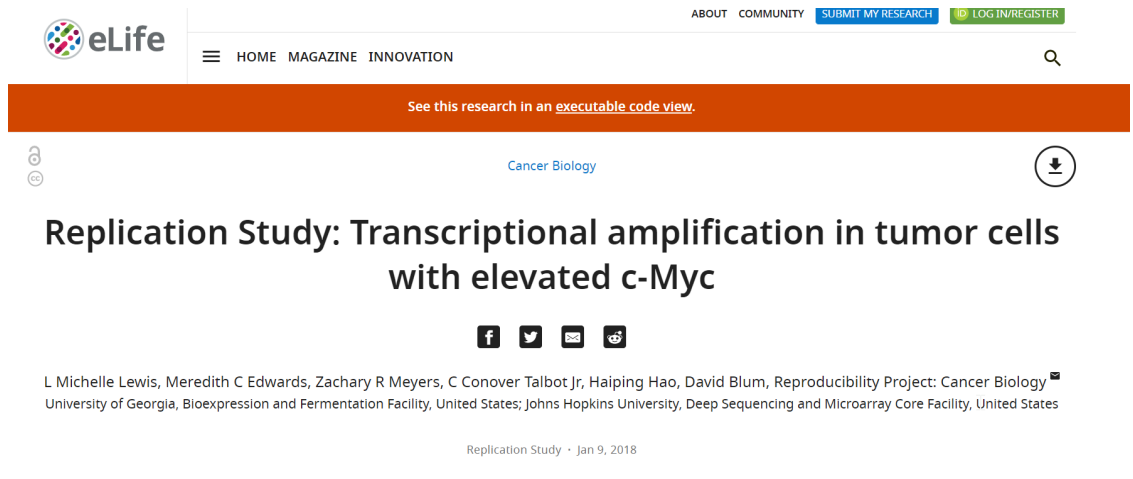


Figure 3.9: UI of the eLife static version [68].

Figure 3.9 shows the original article of eLife which doesn't contain any executable code. Otherwise, this interface allows the user to switch to the executable version of the article through a hyperlink as shown in the figure above.

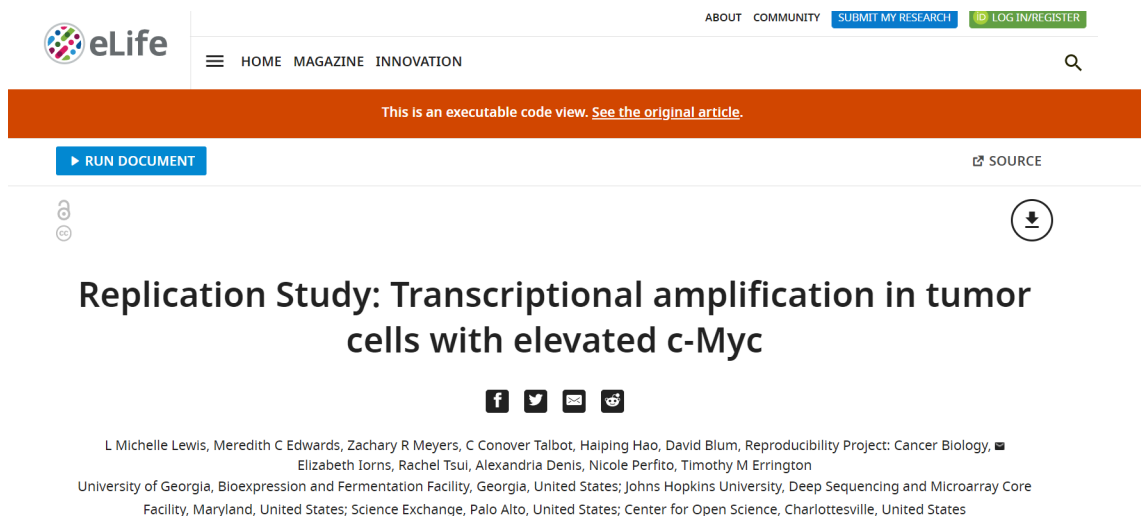


Figure 3.10: UI of the eLife dynamic version [68].

Unlike Figure 3.9, Figure 3.10 contains a *"Run Document"* button which permit to all the code cells integrated in the article. Furthermore, each code cell can be separately executed. A hyperlink permit switching to the original version of the article.

3.4 D4Science: an e-Infrastructure for Virtual Research Environments

3.4.1 Overview

D4Science project (DIstributed colLaboratories Infrastructure on Grid ENabled Technology 4 Science) [69] is a collaboration work of eleven participating organizations of the European Commission's Seventh Framework Programme for Research and Technological Development [70]. It represents the next step of the plan started by Geant [71], EGEE [72] and Diligent [73] projects, and which focus on building networked, grid-based and data-centric e-Infrastructures that boost the multidisciplinary research by removing all the obstacles related to scalability, heterogeneity and sustainability.

Actually, D4Science is nowadays performing an infrastructure composed of various heterogeneous resources which can be classified as follows.

- Hardware resources: contain a set of machines behaving as computing and storage resources providers (mostly used from the EGEE infrastructure) or hosting environment which enable the deployment of dynamic software.
- Software resources: contain a list of software packages that support the implementation of particular functions and services such as the execution of software instances based on functions and data resources.

3.4.2 Concept

The D4Science project is a hybrid infrastructure aimed to ensure the development and application of Virtual Research Environments (VRE) [74] by the as-a-Service provisioning way.

The D4Science-based Virtual Research Environments (VRE) are web-based, collaborative, user-friendly, community-oriented, open science enabler active environment for researchers and experts who cooperate together to attempt a study, a project or a specific research activity. Starting from the end-user viewpoint, each Virtual

Research Environment can be described as a web application containing different components and executing in a simple web browser. VRE users will be able to enjoy the facilities provided by each component. These facilities are developed after including one or more services maintained by several providers. Besides, each VRE aims to play the role of a gateway that organizes the access for the users. For example, this gateway offers the access, to the datasets and services of concern for the selected community and their activities, however, it prohibits the access for the resources providers.

The D4Science can be represented as an IT infrastructure that enables its users to get a comprehensive collection of data management resources offered as-a-Service. These resources represent a set of social networking research facilities that accomplish the infrastructure through offering a scalable list of services that ensure the collaboration among users. Indeed, the users or researchers will be able to share data and news, rate and reproduce data and objects, and interact between each other.

3.4.3 Technology

The D4Science infrastructure is implemented using the gCube software system [75]. gCube can be defined as a distributed system involving Java-based Web Services contributing to the gCube system. Moreover, gCube permits as well the process of large-scale scientific infrastructures. Its design enables it to maintain the entire life-cycle of current scientific inquiry, depending on a certain emphasis on application level specifications of data and knowledge organization. For this reason, it requires the integration of pan-European Grid middleware (gLite) [76] for enabling shared access to a set of exclusive computational and storage resources. Besides, gLite offers as well a large set of services including the collation, presentation, annotation, merging, transformation, search, index and description of information for diverse interdisciplinary and international communities. Indeed, these communities choose, distribute, and consume several infrastructural resources such as information, services, and machines in terms of collaborative Virtual Research Environments.

Additionally, the D4Science Gateway is principally implemented through the use of the Liferay portal technology [77]. This technology is based on a large array of UI components in the form of portlets which are implemented to behave as access points to the fundamental service. In addition to these portlets, a portal has been supplied with extra software components merging it with the rest of D4Science services such as a set of components related to AuthN and AuthZ [78], and other components communicated with the Information System.

3.4.4 UI and Functionality

The D4Science infrastructure is based on many user-interfaces that allow the user to have access to a set of offered features. These user-interfaces can be described in the form of platforms.

The workspace platform

Through the workspace platform UI, the VRE users have the possibility to manage their data and get access to the data shared with other users.

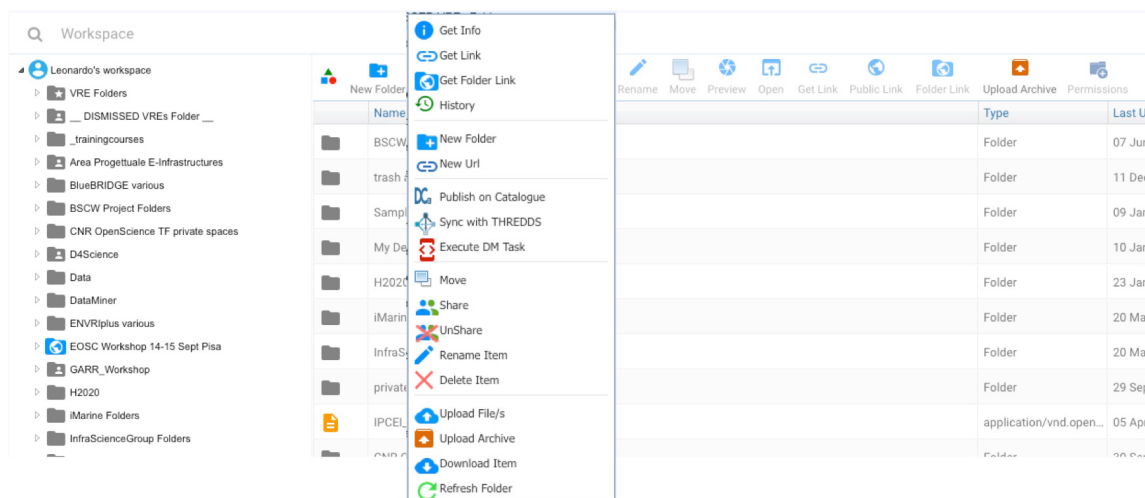


Figure 3.11: The D4Science Workspace platform [79].

Figure 4.14 describes the user interface of the workspace facility which contains a common file system with files organized in the form of folders. According to the figure above, the D4Science infrastructure maintains an unlimited range of items that contain huge and extensible metadata. Indeed, these are stored using an array of storage solutions [80].

The social networking collaborative platform

In addition to the workspace user-interface, the D4Science offers as well an interface for the social networking area. This interface is displayed on Figure 3.12 and permits the communication between the VRE users and their VRE colleagues. In fact, it allows them to receive every information about the accomplishments, discussions

and opinions. Besides, this UI is represented in the form of a social networking area which contains posts, mentions, comments and reactions. Therefore, the researchers can use it as a strong and flexible communication tool.

The screenshot displays the D4Science social networking platform interface. It features a sidebar on the left with sections for 'Statistics' (showing user stats for AGINFRAplus), 'Recent Documents' (listing PDF files), and 'Top Topics' (listing hashtags like #slidedeck, #rstudio, #workspace, #nasa, #godan, #kickoff). The main content area includes a 'Share updates' section with a text input field and a 'Share' button. Below this is a 'News feed' showing a post by Leonardo Candela about RStudio in a VRE, with a link to a wiki page and a comment by Matthias Filter.

Figure 3.12: The D4Science social networking platform [79].

The data analytics platform

The data analytics platform offers the users the possibility to follow and execute their analytical tasks. The user-interface described in Figure 3.13 shows the standalone statistical framework named DataMiner which includes a set of ready-to-use algorithms and functions. Basically, this user-interface defines the heterogeneity and the distribution offered by the D4Science which permits the execution of complex tasks. The DataMiner Master, shown in the figure below, is a web service points to accept requests for running workflows and executing the processes. The execution can be performed locally if the processes depend on local algorithms, or by integrating the DataMiner Worker if the processes are based on distributed algorithms.

3.4. D4Science: an e-Infrastructure for Virtual Research Environments

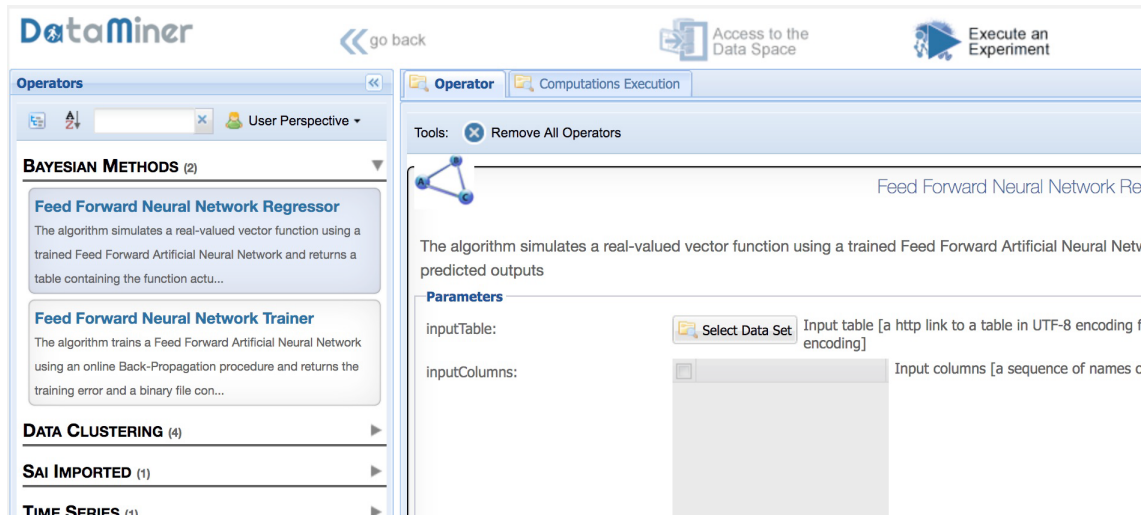


Figure 3.13: The D4Science data analytics platform [79].

The publishing platform

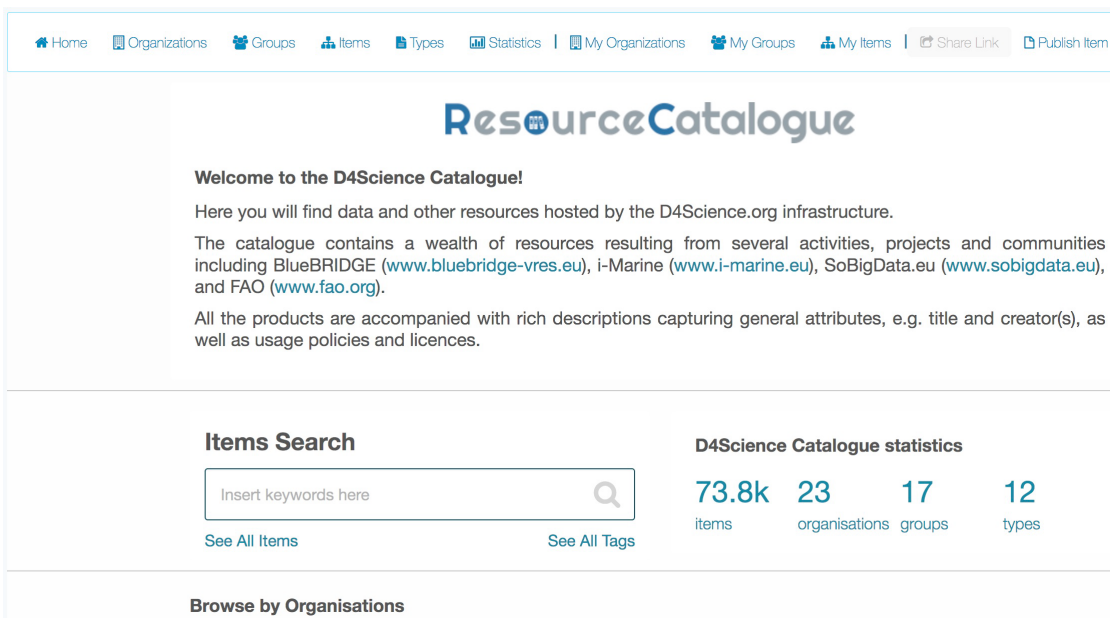


Figure 3.14: The D4Science data publishing platform [79].

Figure 3.14 highlights the UI of the publishing framework. This user-interface allows the VRE users to publish and receive notifications about the availability of particular artifacts at various development level. It consists of a collection of artifacts including search and browse. In fact, the D4Science data publishing user-interface ensures the accessibility to the typologies of the shared products and the metadata of the documents. The integration of this user-interface with other interfaces makes from the D4Science a flexible infrastructure.

Moreover, each published item in the collection of the artifacts is designed by a type which describes its features and enables an easier search. Additionally, every item is characterized by an unlimited range of metadata which permits the description of the item, and an optional resource which outlines the current payload of the item.

Chapter 4

Approach

This chapter highlights the proposed approach of this work with describing the different components that are used to implement this approach. Furthermore, the proposed solution will be thoroughly explained from the five ODP viewpoints.

4.1 Overview of the system

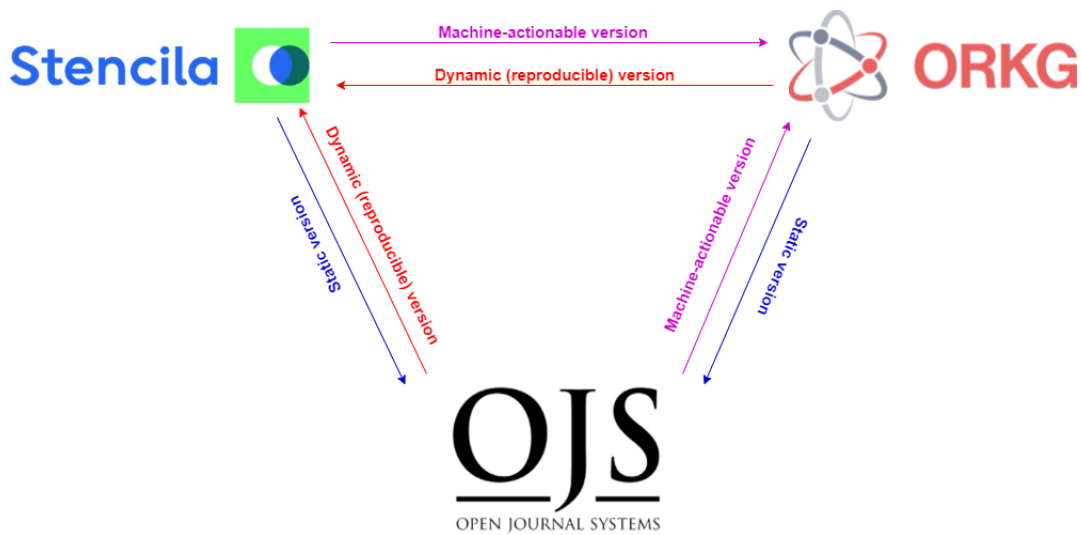


Figure 4.1: The main components of the system and their interactions

Figure 4.1 describes the system of the thesis by stating the three components and the interactions between them. According to this figure, we have adopted Stencila [14] as a framework to ensure reproducibility of the data, the Open Research Knowledge Graph (ORKG) [15] to get a machine-actionable form of the data, and the Open Journal System (OJS) [16] as a traditional publishing framework to submit online our scholarly knowledge. All these platforms are describing in detail in the following section.

4.1.1 Description of the used platforms

Stencila

Overview

Stencila is an open source framework created for researchers. It aims to improve the research literature by enabling scientists in reproducing their publications. It enables also the authoring of interactive, data-driven publications in visual UI [14]. Furthermore, Stencila Desktop helps researchers with limited software knowledge skills by allowing them to use some programming languages such as Python and R within common word process and spreadsheet user interface in order to avoid any complexity of reproducible research.

Features

As mentioned earlier, Stencila is a platform that allows authoring, collaborating on, and publishing executable literature [81]. The core of Stencila is based on various open source packages, developed in several programming languages. Stencila is also available as a Hub [82] on the web, providing thus easy entry for beginners. All Stencila software is available in the Hub through an interactive browser based interface.

Technologies

Stencila offers the users a wide choice of technologies. He/she can use Stencila from within his/her favorite programming language. These packages are in an early, proof-of-concept state and are probably to be implemented further only as the requirement arises. Besides, Stencila provides diverse packages for many languages including JavaScript [83], Python [84], R [85], Rust [86], etc. Stencila permits the possibility to delegate to plugins, but they are only reachable through specific functions such as execute, convert, etc. For example, for installing Stencila via JavaScript or TypeScript [87], the Stencila Node.js [88] package is available from NPM as follows.

```
npm install stencila
```

4.1. Overview of the system

Several plugins are used to run Stencila. They can be installed via the Stencila CLI using the following command.

```
stencila plugins install <name or alias>
```

Table 4.1 highlights the principal plugins used at different levels of development and served to run Stencila. However, only 90% of Stencila environment will be covered when the user install them due to some issue of compatibility with the CLI version.

Plugin	Aliases	Version	Coverage	Primary functionality
encoda	converter	v0.109.2	87%	Convert documents between file formats
jesta	node, javascript, js	v1.8.1	83%	Compile, build and execute documents that use JavaScript
rasta	R	v0.10.2	84%	Compile, build and execute documents that use R
pyla	Python	v0.3.1	87%	Compile, build and execute documents that use Python
jupita	Jupyter	v0.2.4	96%	Execute documents that use Jupyter kernels
dockta	Docker	v0.25.0	79%	Build Docker images for executable documents
nixtata	Nix	v0.1.2	14%	Build Nix environments for executable documents

Table 4.1: Stencila plugin list [81].

UI and functionalities of Stencila Hub

Stencila provides user accounts for individuals or for enterprises (organizations). There are two ways of registration, either through an existing account of GitHub, Google, ORCID, and Twitter, or through an email address and a username. After registration, the user can log in and get access to his/her Dashboard where he/she can personalize his/her profile and edit his/her information described in settings as shown in Figure 4.2.

The screenshot shows the Stencila user profile page for 'Anouar Ganfoud'. The page is divided into a left sidebar and a main content area. The sidebar contains a list of navigation items: Projects, Settings, Profile (highlighted), Publishing, Password, Emails, Connections, Invites, Features & privacy, and Plan. The main content area is titled 'Profile' and includes the following fields:

- Name:** anouar (with a note: 'Name of the account. Lowercase and no spaces or leading numbers. Will be used in URLs e.g. https://hub.stencila.io/awesome-org')
- First name:** Anouar (with a note: 'Your first names (given names)')
- Last name:** Ganfoud (with a note: 'Your last names (family names, surnames)')
- Location:** (with a note: 'Location to display in account profile')
- Website:** (with a note: 'URL to display in account profile')
- Email (public):** (with a note: 'An email to display in account profile. Will not be used by Stencila to contact you')

An 'Update' button is located at the bottom of the form.

Figure 4.2: Stencila personal page of the user.

Every registered user has the possibility to create a project under his/her private account or under his/her organization. An organization project allows all the associated users to work cooperatively on this project. Once the project is created, the author can choose the theme that he/she plans to work on, then he/she can upload files and add it to the project in order to use it within the project executable documents. These files do not require to be downloaded because they will be managed on Stencila servers. Further, the user can also add sources to his/her project that are hosted in another server. They will persist on this external server, but a versioned copy is pulled and saved in the project to guarantee reproducibility. Once the session is computed, the files will be also pulled into this session. When the user changes his/her code and want to push the changes to Stencila, he/she can access to the source tab and update the local copies of his/her code files. Besides, it is possible to convert the files to different formats (Microsoft Word, (.docx), HTML(.html), JATS XML(.jats.xml), JSON(.jsonId), R Markdown(.rmd), Jupyter Notebook(.ipynb), YAML().yaml, etc.) to implement other tasks as shown in Figure 4.3. In order to make the document executable, it is mandatory to select the file that you want to publish it as a main file. However, every file can be automatically picked as main file when it has the name *"ReadMe"* or *"Main"*. The main file can be considered as the home page of the project which enables the users to show the

4.1. Overview of the system

document.

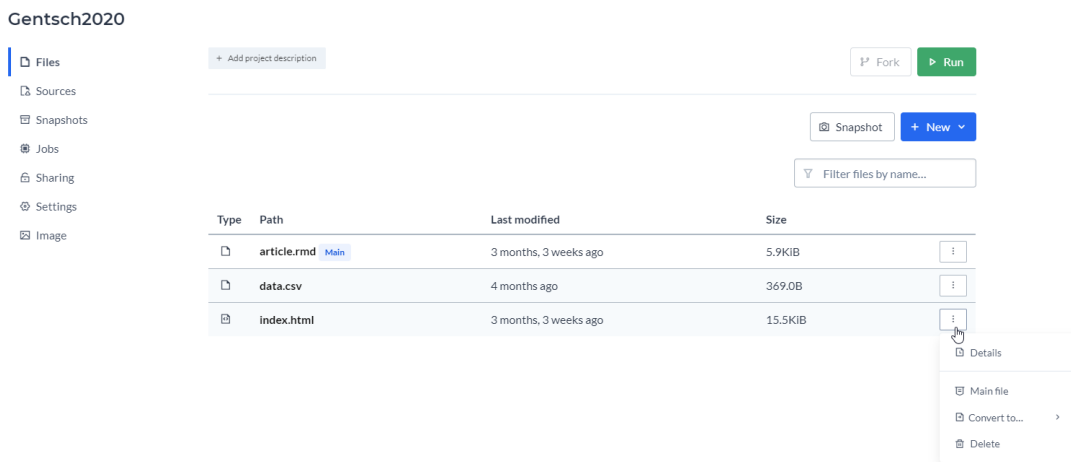


Figure 4.3: Stencila files management page.

After choosing the main file, the user can create a snapshot via the button shown in Figure 4.4 to generate an HTML file through the converting of the main file. Indeed, the snapshot set up a capture of the saved version by building a copy of all the project file exactly at that particular time. Thus, after every upload or change in the source file, a snapshot should be taken to record the new changes. At the end, a user may have a long list of snapshots ordered sequentially.

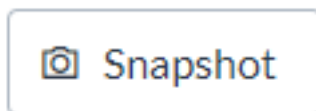


Figure 4.4: Stencila Snapshot Button.

A preview of the executable document is ready to be shown after converting the source files to a modifiable manuscript. Before submitting the document for publishing, the user can download it to his/her local machine or open it and replace any static figure in the paper with Code Chunks. The user can also share this document by giving the generated URL to his/her collaborators. The URL is unique and specific for each project because it contains its name. Thus, changing the name or the title of a project will affect the URL.

ORKG

Overview

The Open Research Knowledge Graph (ORKG) is an open source project developed by the TIB (Technische Informationsbibliothek) organization to push researchers to participate in the improvement of technologies and uses cases for open graphs for research [89]. This project plays an important role in advancing scholarly communication. Indeed, ORKG allows authors to define their papers in a structured way represented in knowledge graphs [15]. These graphs describe scholarly knowledge included in the literature. In addition to bibliographical metadata, ORKG encloses machine actionable representations of academic knowledge. Basically, the ORKG project facilitates the task of researchers by making the papers easier to find and compare.

Features

ORKG represents a beneficial tool for scientists thanks to many offered features such as:

- Research contributions can be perfectly structured: Users can overcome the traditional description of the research contributions and use the ORKG to describe in a structured and semantic way [90].
- Researchers can create templates that define the structure of content types: These templates can be later adopted to describe research contributions to avoid ambiguous semantics.
- The possibility of comparing the contributions: ORKG allows researchers to perform a comparison between contributions concerning a particular problem. Several examples of comparisons can be identified in computer science such as the comparing between the best, average or worst case performance of sorting algorithms.
- ORKG permit the visualizations of graphs: As its name mentions, ORKG allows users to convert their articles and researcher contributions to a highly interacted user interface in the form of a graph.
- Papers and contributions will be organized by experts: ORKG has a group of experts called Observatories which are responsible to inspect and organize ORKG data in a way that it respects some predefined rules and disciplines.

- Reproducibility is improved thanks to ORKG: Scholarly knowledge and comparisons of different contributions encourage scientists to reuse the data easily.

Technologies

ORKG python library [91] is simple to implement and can be used by any user even though users with limited programming knowledge. Only a basic knowledge of python and JSON is required to get involved in the project. The ORKG package can be used to manipulate (add, modify, consult) data from any instance of the open research knowledge graph. Basically, the data science process requires this data to fetch it for analysis and visualizations.

The installation of the ORKG package can be performed by cloning the repository from GitHub or through the package installer for python *pip* as:

```
$ pip install orkg
```

Once the user has installed the ORKG package, he/she can import it to his/her python code. He/she needs also to create an instance of the main class to use it:

```
from orkg import ORKG # import base class from package
```

```
orkg = ORKG(host="<host-address>", creds=('email', 'password'))
```

Users credentials are necessary to establish authentication to the user account. Indeed, the ORKG package allows the connection to any local or remote instance of ORKG.

Furthermore, the ORKG python package needs two principals classes:

- ORKG class which enables users to connect to an ORKG reference and contains the output of all request in the ORKG python package.
- ORKG Response which encloses the responsible output for the ORKG API. It encapsulates the data about the request and response from the destined API.

In addition to these two classes, ORKG package provides other components, in particular the ORKG resources, predicates, literals and objects.

UI and functionalities

A variety of user-interfaces offer to users the opportunity to describe their research data in a structured manner. Figure 4.5 demonstrates the structured description of a

research contribution for enhancing osteoblastogenesis by functionalizing PCL scaffolds with anti-miRs. In addition to the addressed problem, the research contribution describes the utilized materials and methods, and the acquired result.

The screenshot displays the ORKG interface for a research contribution. At the top, there are two tabs: 'anti-miR functionalized PCL scaffolds' (highlighted in red) and 'Heterotrophic Bone Formation'. Below the tabs are three filters: 'miRNA expression / RT-qPCR', 'miRNA targets', and 'small RNA-Seq'. The main content area is titled 'Research problems' and includes a checkbox for 'Add to comparison'. The research problem is 'enhancing osteoblastogenesis by functionalizing PCL scaffolds with anti-miRs'. Under 'Contribution data', there are three sections: 'Has material' (anti-miR of hBMSC-TERT, Polycaprolactone), 'Has method' (Alizarin Red S staining, ALP activity, anti-miR scaffold functionalization, DAPI staining, HA staining), and 'Has result' (increased ALP activity, increased calcium content).

Category	Value
Research problems	enhancing osteoblastogenesis by functionalizing PCL scaffolds with anti-miRs
Has material	anti-miR of hBMSC-TERT Polycaprolactone
Has method	Alizarin Red S staining ALP activity anti-miR scaffold functionalization DAPI staining HA staining
Has result	increased ALP activity increased calcium content

Figure 4.5: ORKG Interface of the research contribution.

After displaying the contribution data, the user can compare it with other contributions that discuss the same research problem. Figure 4.6 depicts a comparison of the transmission potential of COVID-19 between a collection of countries such as Italy and Iran. This comparison permit the concerned scientists to evaluate the

4.1. Overview of the system

different data of this virus like the date of the virus beginning, the period, the basic reproduction number, the confidence interval, the used method, the lower confidence limit and the upper confidence limit.

Properties	The early phase of the COVID-19 outbreak in Lombardy, Italy 2020 - Contribution 1	Transmission potential of COVID-19 in Iran 2020 - Contribution 1	Transmission potential of COVID-19 in Iran 2020 - Contribution 2
Location	Lombardy, Italy	Iran	Iran
Time period	Time interval	Time interval	Time interval
Has beginning	2020-01-14	2020-02-19	2020-02-19
Has end	2020-03-08	2020-02-29	2020-02-29
Basic reproduction number	Basic reproduction number estimate value specification	Basic reproduction number estimate value specification	Basic reproduction number estimate value specification
Has value	3.1	3.6	3.58
Confidence interval (95%)	Confidence interval (95%)	Confidence interval (95%)	Confidence interval (95%)
Lower confidence limit	2.9	3.4	1.29
Upper confidence limit	3.2	4.2	8.46
Method*		generalized growth model	based on the calculation of the epidemic's doubling times: estimated epidemic doubling time of 1.20 (95% CI, 1.05, 1.44) days

Figure 4.6: ORKG Interface for comparison between contribution.

ORKG allows users to specify the structure of the type composition to describe research contributions. Templates are used to facilitate the organization of data by adding, editing or deleting attributes and their allocated information. For example, Figure 4.7 highlights the specification of the features of describing a dataset.

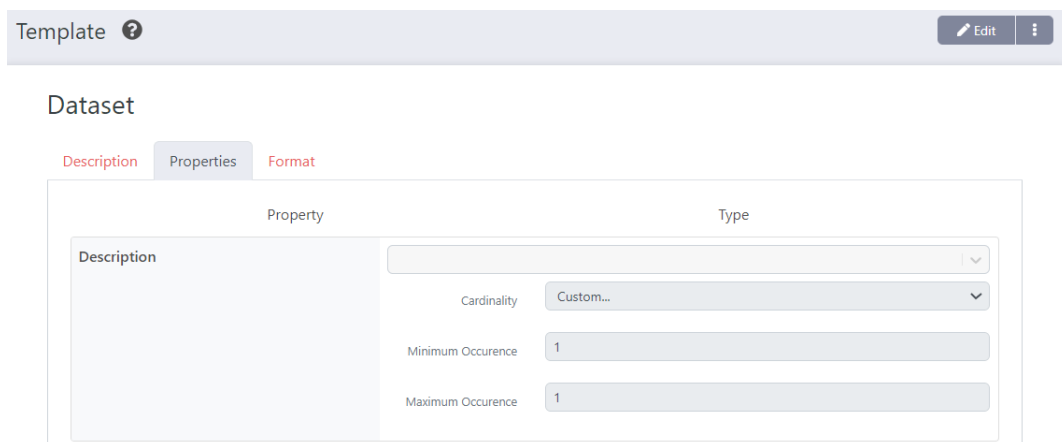


Figure 4.7: ORKG Interface of a Dataset template.

Via the button *"Graph View"*, the user can visualize his/her paper in the form of a graph that describe the contributors, the research problem, the publication data and the research contribution as shown in Figure 4.9.

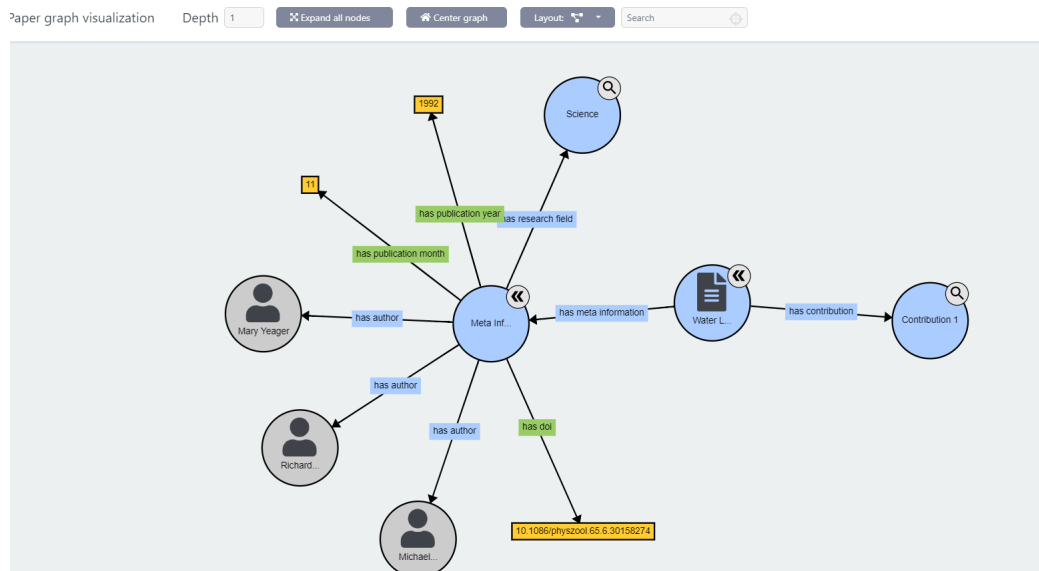


Figure 4.8: Visualization of a paper in the form of a graph in ORKG.

OJS

Overview

The open journal systems (OJS) is an open source framework aims to manage and publish academic literature online in order to avoid expensive traditional publishing approaches such as printing [16]. OJS is available for free download and can be installed on a local machine or on a web server. It allows the administration and sharing of scholarly journals in a flexible manner. This platform was first launched in 2002 as part of Public Knowledge Project's (PKP) research project led by John Willinsky at the University of British Columbia (UBC) in Vancouver, Canada [92]. The purpose of OJS inventing was firstly to help print journals to save up money by publishing their content online. Then, this framework is used to manage the journal process, from document submission based on review to reporting and publication. Besides, authors will be able to make their publications freely available on several online institutional archives and the journal will be then freely reachable to readers.

Additionally, the open journal system represents the most largely used open source journal publishing project in existence, with more than 10,000 journals using it worldwide [93]. This platform points to improve the scholarly and public feature of the concerned research based on its administration systems, its highly flexible indexing of research, and the supplied context for the study. Basically, the journal readership and its contribution to the public can be enhanced thanks to the open access feature. Furthermore, OJS offers the possibility to add automatically the submission of DOIs and to control the reviews and the publishing of completed documents [94].

Actually, the administration of OJS journal is not an easy task. A user should have enough knowledge and a respectful level of management skills to install and configure correctly the system, and resolve the potential issues. Even though that OJS is an open free system adopted for open access journals, charging access costs and managing subscribers are subsidized by the OJS platform.

Features

OJS represents a comprehensive platform for the administration of the hole submission and editorial process and sharing the documents online with the readers thanks to these features:

- The Workflow is responsive and can be configured
- The entire content can be online managed
- Possibility of installation in local machine and web server

- Flexible user front-end with the ability to choose the theme and the design
- Academic publishing services such as Crossref, ORCID, and DOAJ are supported
- Simple indexing and easy discoverability which makes it recommended by many research platforms like Google Scholar
- It Supports more than 30 languages
- The registration module is enriched with several open access options
- Advanced documentation with expanded user guides and many training videos
- A favorable support and lead provided by the community

Technologies

For the implementation of the version 2.x of the open journal system, the object-oriented framework PHP is adopted with the integration of the Smarty template system for user interface abstraction. SQL database is also used to store the data via abstracted database calls performed by the ADODB Database Abstraction library. Thus, in order to install OJS on a local machine or on a local web server, these configurations are required:

- PHP framework (the version 4.2.x or later)
- MySQL database (the version 3.23.23 or later). Otherwise, the PostgreSQL (the version 7.1 or later) can be also used.
- Apache Server (the version 1.3.2x or later). Alternatively, the Apache 2 (the version 2.0.4x or later) can be adopted or the Microsoft IIS 6. However, this version of Microsoft IIS requires the installation of PHP 5.x.
- As operating systems, almost the most used OS can be adopted such as Microsoft Windows, Mac OS X, Linux, BSD, Solaris.

In addition to all these systems, many other versions or frameworks can be also used but are not supported and may not have been already tested. Feedback from users is always checked by the community to improve the OJS configuration and inspect which platforms not listed above have successfully run the open journal system.

UI and functionalities

Simple, functional and responsive user interface is offered by OJS to permit users to easily access, configure, review and publish online their documents. Figure 4.9

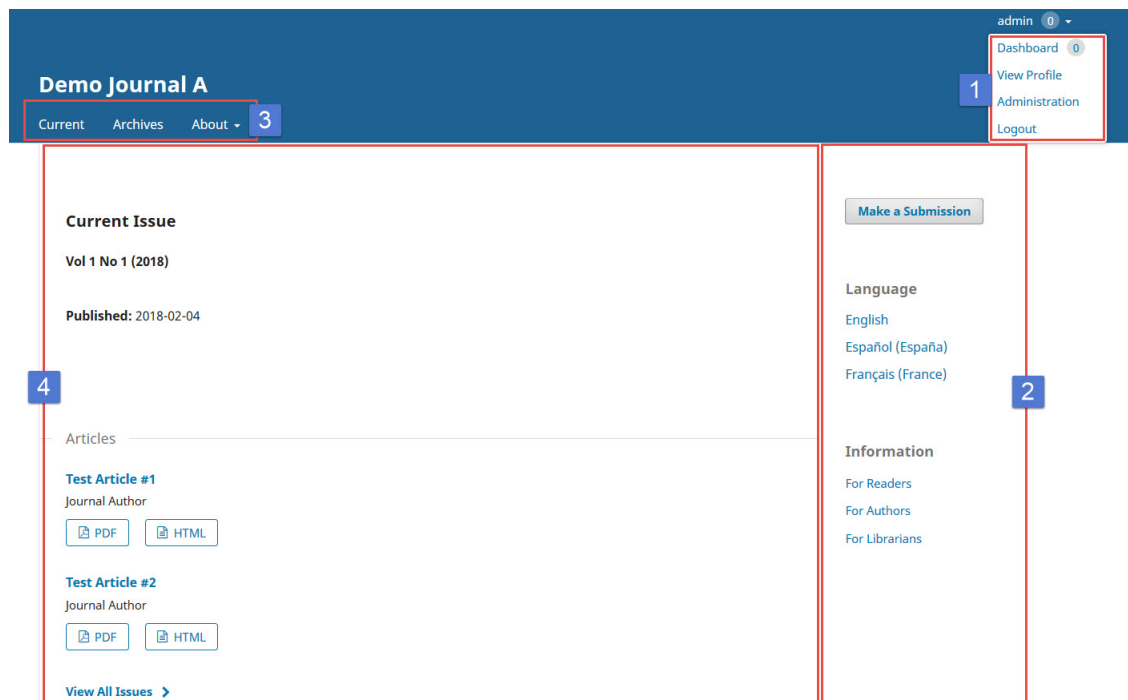


Figure 4.9: OJS Reader Interface [95].

describes the user functions from the profile menu (1). The creation of a submission or the switching between profile interfaces and languages can be done through the side navigation bar (2). However, the top navigation bar (3) encapsulates the collapsible menus for the “About” features. The information of the article such the linked title for accessing to the object metadata and abstracts are described in the main content block in the middle of the page (4). OJS 3.x allows the user to log into their editorial systems where the reader can access to several tabs as shown in Figure 4.10. This dashboard interface enables the user to see different information about the worked on content via the main panel sub-menus (5). The right panel (6) contains the action buttons that represents a schedule for publication.

Jupyter Book

During our thesis, we tested and analyzed Jupyter Book, as well as Stencila, as a framework that ensures the reproducibility of scholarly knowledge. However, we

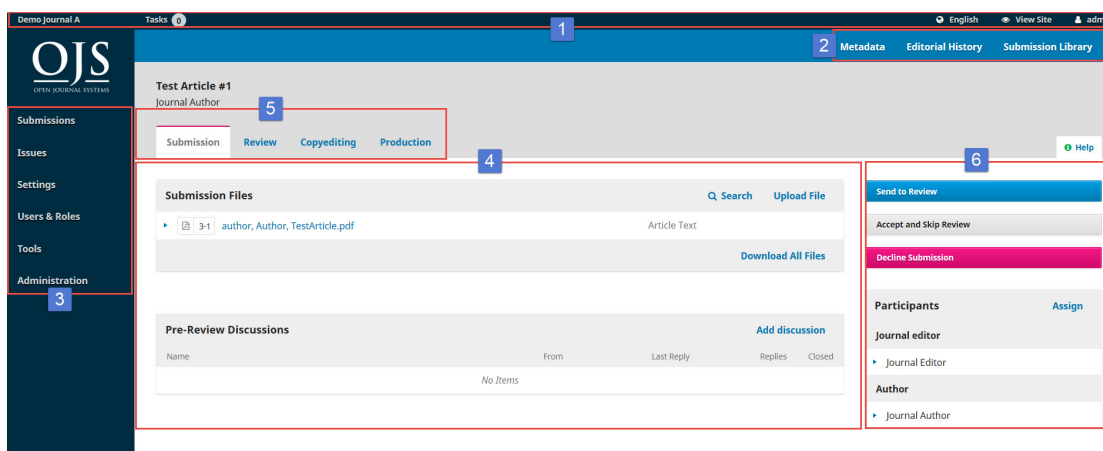


Figure 4.10: Dashboard interface in OJS 3.x [95].

decided to use Stencila as a platform for our work for many reasons such as its capability of reproducing individual values, its simple installation and configuration, etc.

Overview

Jupyter Book is an open source platform designed for creating elegant, publication-quality literature such as books and documents from standard computational and data science materials such as Jupyter Notebooks and Markdown scripts [96]. It represents a part of the Executable Book Project, which is an *“international collaboration to build open source tools that facilitate publishing computational narratives using the Jupyter ecosystem”*. Its last version is v0.10.1 and is always being updated. There is a big large community of Jupyter Notebook on GitHub that is daily growing to contribute and discuss its issues.

Features

Jupyter Book aims to improve reproducible science by offering researchers and authors several features:

- The possibility to write publication-quality content including plots or figures, mathematical calculation, cross-references and citations: The user can use either Jupyter Markdown or an advanced version of Markdown for writing his/her publications.
- The content is written in Jupyter Notebook: The author can integrate his/her code and outputs into the book. Jupyter Book allows also the user to write

notebooks completely in Markdown which will be executed during the build of the book.

- The document content can be executable and cached: Jupyter Book offers the scientists the possibility to run their code and add the latest outputs into the article. Besides, the user can also cache his/her results to be adopted later.
- Enabling the including of notebook outputs into the user content: The author is able to produce the outputs of his/her code and insert them into his/her book pages.
- Including Interactivity to the book: Integrate interactive outputs like plots and widgets, linking to an online service such as Binder and adjust the visibility of cells.
- Different outputs can be generated: such as websites (HTML, CSS, JavaScript), Markdown and PDF.
- Easy way to build books: Generating books can be performed via a simple command line which is:

```
jupyter-book build mybook/
```

Technologies

Jupyter Book is based on a collection of technologies in the Python ecosystem that facilitates the publishing of computational literature. In fact, Jupyter Book uses a MySt Markdown language in Markdown and Notebook for enabling authors to write heavy, publication-quality markup in their papers. Additionally, a MyST-NB package is required to perform the parsing and the reading of notebooks. In order to generate outputs from the content of the book, Jupyter Book adopts as well the Sphinx documentation tool engine. Otherwise, various Sphinx plugins and tools are used to allow the users to add new features.

UI and functionalities

Jupyter Book permit the users to integrate computational data in their book. Figure 4.11 shows the insert of the code chunk in order to manipulate the data used to generate the plot.

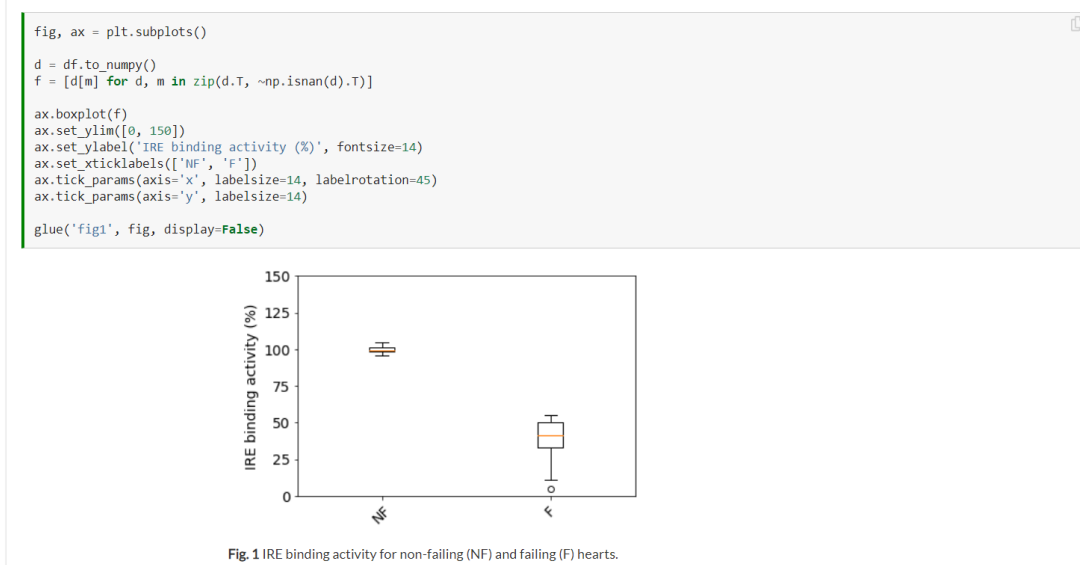


Figure 4.11: Generating plot with code chunk in Jupyter Book [97].

Jupyter Book gives the possibility to hide some code in the book via the tag “*hide-input*” to the responsible cell in Jupyter Lab as described in the figure below.

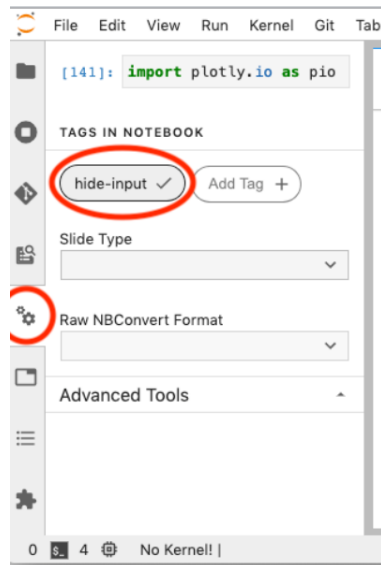


Figure 4.12: Adding the “hide-content” tag to a cell in Jupyter Lab.

4.1. Overview of the system

Basically, Jupyter Book is based on Jupyter Notebooks, this allows users to launch live Jupyter sessions in the cloud and interact with the code chunk. Four platforms are connected to Jupyter Book in order to provide interactivity as described in Figure 4.13: JupyterHub, BinderHub [65], ThebeLab [98] and Google Colab [99]. For instance, BinderHub can be adopted to create the environment required to execute a repository, and supplies a link that allows other users to interact with that repository. If Jupyter Book is hosted online on GitHub, the user is able to automatically add buttons that establish the connection to the Jupyter Notebook executing in a BinderHub. When the user clicks the button, it will be redirected to a live version of the page.

In order to automatically include Binder link buttons in each page of the Jupyter Book, it only remains to add the following configuration.

```
$ use_binder_button: true
```

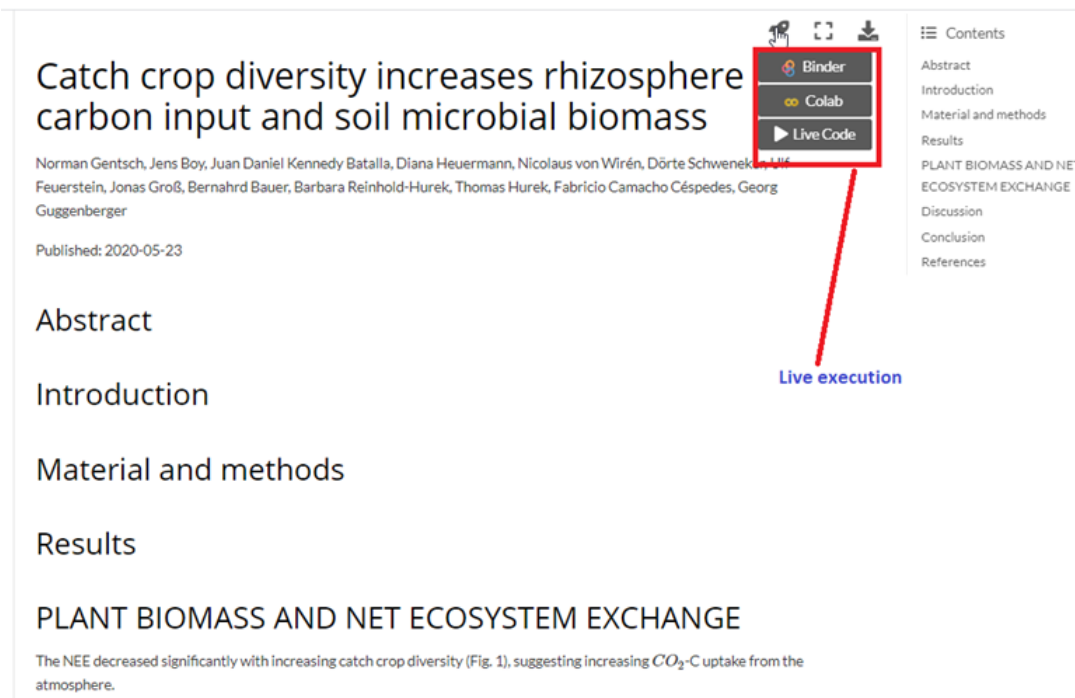


Figure 4.13: Adding interactivity through Binder, ThebeLab and Google Colab.

4.1.2 Description of the system from the ODP viewpoints

After selecting the components of our system, we choose the reference model ODP as an architecture framework to define the structuring of our system specifications. Figure 4.14 describes the integration between the three used platforms from the five generic and complementary viewpoints of the ODP reference model.

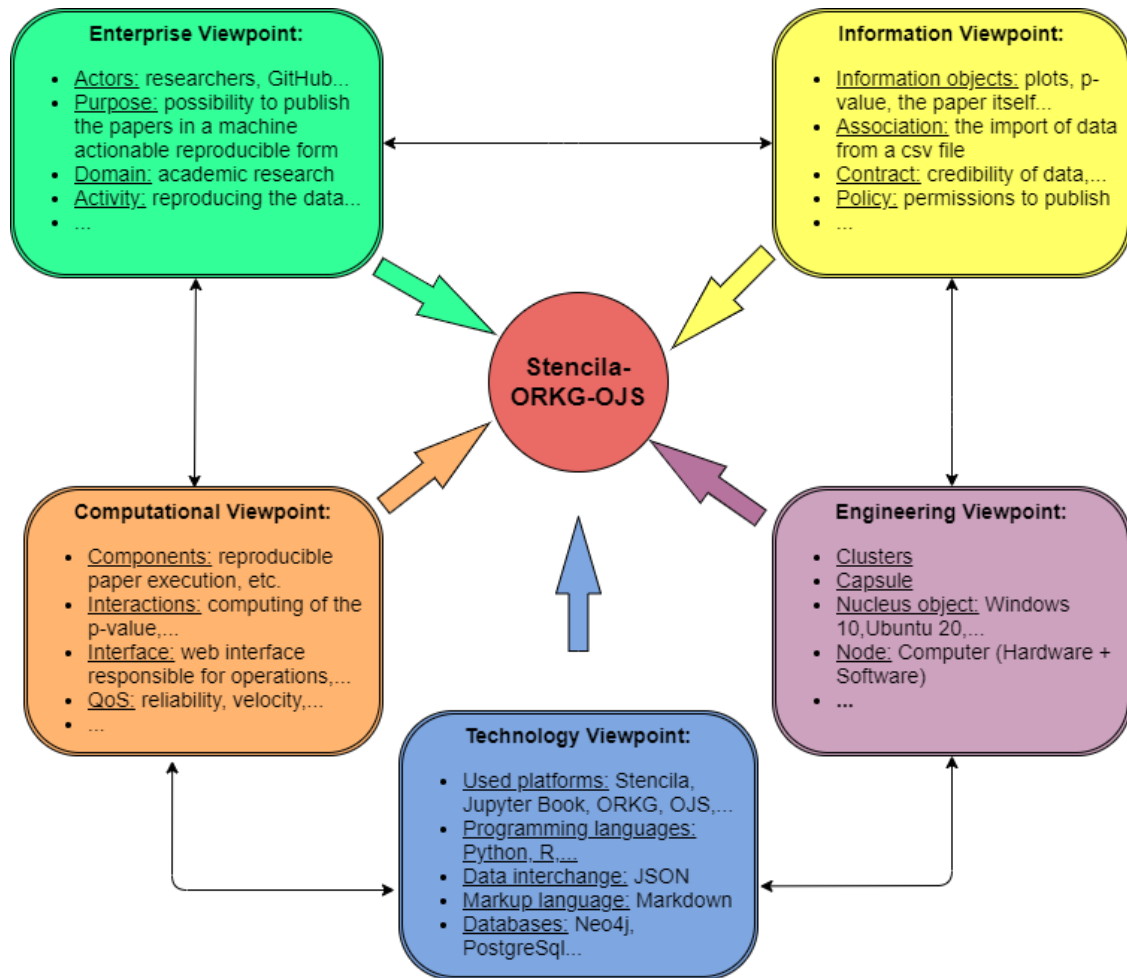


Figure 4.14: The Stencila-ORKG-OJS system of our study

As stated in Figure 4.14, each ODP viewpoint of our system consists of various key concepts described as follows.

Enterprise viewpoint

The key concepts we used in the enterprise viewpoint of our system are:

- **Purpose and targets:** In RM-ODP, this concept is adopted to capture the motivation for the system. The aim of this thesis is to help researchers to publish their publication in a machine-actionable reproducible manner under consideration of the environment. This assumes analysis of the user specifications.
- **Domain:** In the ODP reference model, a domain covers a collection of objects, closely related to each other. Several domains can be studied during our work such as data science, scholarly knowledge and academic research.
- **Activity:** The activity is the actions performed by an enterprise. In our thesis, "*recomputing the data*", "*describing the paper in a structured way*" and "*publishing the paper online*" can be considered as enterprise activities related to our approach.
- **Community:** As highlighted in RM-ODP, a community contains a set of interacting objects whose target is to achieve an objective. They are actually the providers of the platforms and services of a project. During our work, we will communicate with many communities that are related to the tools and the used platforms described previously in detail.
- **Actor and Artifact:** An actor is represented by an enterprise object, a human agent or a software agent, which plays a role in the enterprise view. The user or the consumer can be represented as a human actor. However, a web interface or a data server can be considered as a software actor. In this case, we use the term Artifact. The users or researchers can be defined as actors for our work. Otherwise, the GitHub server can be considered as an artifact.
- **Role:** It represents the unique specification that describes the behaviors of the objects toward the community. "*Publisher*", "*Author*", and "*Reader*" represent the main roles of our study.
- **Scope:** It contains the list of used roles of the system. Basically, the set of roles that describe a business, is the scope of that business. In our Stencila-ORKG-OJS system, we can allocate the scientific publishers, for example, as the scope of the project.
- **Contract:** This concept can be represented as the defined agreement to particular behaviors of the system. The study could be informed of the system

expectation through the contract. In our work, we will define as a contract, the credibility of data carried out from the researchers and shared to other researchers.

- **Policy:** The system should be also constrained by a set of policies that covers the obligation, prohibition, or permission rules that either force or allows actions, as related to the aim. The publication permissions and the security norms could be included in our system as policies. Indeed, we have to give permissions to other researchers to get access to our project. Nevertheless, some restrictions should be defined to protect the work.

Information Viewpoint

The key concepts we used in the information viewpoint of our system are:

- **Information Objects:** This concept is related to the set of objects in the information viewpoint. These objects will be required in the interaction process of our system as well as the objects related to the enterprise viewpoint actors and artifacts. Many information objects are adopted in our system such as the plot and the stated p-value in the paper or more generally the paper itself.
- **Association:** In RM-ODP, an association describes the relationship between the information elements. We require it in every study to determine the connections between the objects. The association between a dynamic platform such as Stencila and a static platform like OJS creates the data reproducibility concept used in the two platforms.
- **Contract:** As mentioned in the enterprise viewpoint, a contract represents the specific agreement to the system objects including the information objects.
- **Policy:** As described in the enterprise viewpoint, the policies represent the set of constraints contained in a system's contract. In our thesis, an example of policy is a security requirement, which have to be established and adopted to allocate all security, required at the suitable and the agreed levels.

Computational Viewpoint

The key concepts we used in the computational viewpoint of our system are:

- **Computational Objects or Components:** This concept covers the set of computational objects or the objects that interact dynamically at interfaces. The

collection of these objects is defined from the objects already described in enterprise and information viewpoints. The reproducible paper execution, creation, storing, curating machine actionable description are examples of computational objects of our system.

- **Interactions:** In our RM-ODP model, an interaction is defined as an activity that includes one or many objects and their environments at a specific interface. Our system interactions state the list of services that are provided by a single interface, and are associated to another object with a defined linking or binding. For example, the calculation of the p-value can be represented as an interaction because it is an action that involves many components such as dataset, methods of computation and methods of visualization.
- **Interface:** Based on the RM-ODP architecture, an interface is related to the behavior of an aspect at a component of the aspect interactions constrained by the circumstances for when they take place. Actually, a computational interface represents a type of interface where the communications have the type of interrogation (request/response) or declaration (publish/subscribe). The code used in our study will interact with the hosting server by sending and receiving a set of requests and responses via a dedicated interface.
- **Binding:** According to the RM-ODP model, a binding can be described as an agreement between two or more object interfaces. This binding represents the results of an agreed-upon behavior. The bindings support the used interfaces and supply the environment where our project interactions will be computed.
- **Quality of Service (QoS):** This concept includes several metrics of Quality of Service. Indeed, these metrics are required to every system computational viewpoint. The required bandwidth, delay and reliability requirements have been adopted in our study as a QoS metrics.

Engineering Viewpoint

The key concepts we used in the Engineering viewpoint of our system are:

- **Cluster:** In RM-ODP, a cluster is composed of a set of related basic engineering objects that will always be connected to each other.
- **Capsule:** The capsule enclose one or more clusters. These clusters will be managed by a cluster manager, and the capsules will be also allocated to a capsule manager.

- Nucleus object: It represents the extended operating system where our ODP is implemented. In our case, we will use both Windows 10 and Linux OS (Ubuntu 20).
- Node: During our project, we have adopted a simple standard computer. It describes the computer system that encapsulate a collection of entities (hardware, software and liveware) that are designed to receive, process, manage the amount of used data.

Technology Viewpoint

The technology viewpoint describes the services and the components provided by the software architecture in a logical level. It involves as well the different alternatives, and determines technological gaps. Several technologies are used during all phases of this work such as some open source frameworks like Stencila, Jupyter Book, ORKG and OJS, or some programming languages like Python and R, databases such as Neo4j and PostgreSQL. We will explain further all these technologies in detail in the next chapter of this work. The installation, requirements and configuration of the used frameworks will be as well described.

4.2 Description of the proposed solution

To face the challenges of research literature, it becomes important to ensure the reproducibility of machine-actionable expressions of essential scholarly knowledge. We will try to avoid the aspects of the conduct of research that contribute to irreproducible results, and follow the suitable strategies for disseminating good practice in this area. As a proposed approach, we suggest, as the first step, to create articles that contain static data in a dynamic way in order to permit other users to see the data and recompute it. In a second step, we will try to integrate the computed work with other scholarly knowledge frameworks such as the open research knowledge graph (ORKG) and the open journal systems (OJS). Indeed, as we used Stencila for the reproducibility part, we will use ORKG for the machine actionability part and OJS for the traditional publication as described in Figure 4.1.

4.2.1 Creation of executable articles

During this phase, we will focus on finding a way to help scientists to openly share results and the underlying data with other scientists. Therefore, after studying the

potential tools that supports reproducible science, we decide to test Stencila [14] and Jupyter Book [96] as host frameworks for the computational work.

In Stencila, we create a new project, and we integrate the data of a pre-selected paper into its sources files. The article information such as contributors, publisher, publication data should be written in R markdown. Further, if the article has a dataset, we can put it into a CSV file in order to allow users to separately review it and edit it. This CSV file will be later added as a dependent file to the Stencila project. However, the code responsible for the generation of a plot, table, p-value or any computed item will be involved with the article Markdown text in the form of Stencila code chunk. Thereafter, it only remains to publish the project via the Stencila hub and allows different users to see and compute the related data.

As well as Stencila, we have tried to test the reproducibility of articles using Jupyter Book. We have used Anaconda Navigator [100] as a desktop graphical user interface (GUI) to launch our application before we host it online. As mentioned in previously, Jupyter Book content should be written in Jupyter Notebook. However, the code of the plot is written in R. This was the first challenge that we faced during the use of Jupyter Book. Luckily, Anaconda supports the integration of R code in a Jupyter Notebook after the including of an R kernel. We put then some code that permits us to visualize a plot in a separate cell in the form of a code chunk as well as another code responsible for the description of the p-values table. Thus, after creating the article and its dependent code and data, it is still only to publish it online via GitHub in order to offer to other users the possibility to run it.

Once we complete testing Stencila and Jupyter Book, it remains only to make the final decision and select the suitable one between them. We chose Stencila as our reproducible framework since it is a new platform that has some advantages comparing to Jupyter Book, such as the possibility to reproduce the individual values. Therefore, we will test its integration with ORKG and OJS.

4.2.2 Integration

In order to maximize the availability and the efficiency of our frameworks (Stencila, ORKG, OJS) and avoid any issue of incompatibility or dysfunction after regular updates, it is preferable to install all these platforms locally before we start its integration.

Integration between Stencila and ORKG

Once we have hosted our platforms, and we have created an executable paper in Stencila, we will simulate how ORKG can harvest data from Stencila projects. There-

fore, we firstly should import the dataset related to the article into ORKG using the python library destined for that. After importing the data, we will get an ORKG resource ID that can be used for reading the data through the ORKG API using R. However, we don't need to do that since we already have the CSV file in our Stencila project. Since the paper is published, it will be favorable to cache the data together with the paper and avoid reading it directly from the ORKG because it is too fragile against changes. Once we have the ORKG resource ID for the dataset, we will create a machine-readable description of the hypothesis test. In this description, the resource ID for the dataset will be an object and the test has the dataset as an input. Hence, the easiest way to describe this machine-readable description is adopting the ORKG Python library native format represented in JSON. Based on the ORKG documentation, we can also use the ORKG terms such as resource and predicate IDs. Each described in ORKG may have one or many contributions which can contains a dataset as material for the article. In addition to the dataset, the contribution can enclose one or different results. Therefore, we will create a JSON file through the call of the required function in R/Python to which we can give the dataset and the result and then the function creates the machine-actionable description of this information as a statistical hypothesis test. Thereafter, we include this JSON file into our Stencila project. Finally, it only remains to write a Python script that permits the adding of our paper to an ORKG instance.

Integration between Stencila and OJS

After installing OJS locally, we will be able to establish a journal and have access to all the operational tasks from creating an author's submission to the peer review, the editing, publication, archiving and the indexing of the journal. Based on the example of the integration between eLife Magazine and Stencila explained in the related works chapter, we will try to build a similar integration between OJS and Stencila that allows the users to jump between a static version of the paper created in OJS and a dynamic executable version developed in Stencila. We will again select a random paper. Firstly, we should rewrite it in OJS. Therefore, we need to create an issue that encapsulates our paper. This issue represents the volume matriculate and number of the journal. Thereafter, we will be able to create our submission that contains the title of the journal, contributors and many other optional related information such as publication date, research field, etc. Once we have created our journal, we can change the layout of the paper to make the output of the journal page similar to what we get in Stencila. The easiest way to do that is to download the HTML file responsible for the paper output in Stencila and edit it by rejecting

all the dynamic items (code and data) and replace it by HTML calls for uploaded static data. Furthermore, we will add this HTML file as a OJS galley with all the dependent files such as a screenshot of a figure. Another galley will be needed to establish the switch to the Stencila executable version. This galley will contain an HTML file that encapsulates the output of Stencila in a "*HTML iframe*" element and a link that allows the jump between the static and dynamic instance. However, Stencila won't enable any foreign framework to load its projects without permission. Hence, before we publish the journal, it is necessary that we configure the publication settings of the selected paper in Stencila to give access to OJS to load it. Once all the galleys are added, we can publish our journal, certainly after publishing the issue that enclose it. Since, the journal is published, the user will be able to view as an online website and then to try the integration between the two frameworks.

As described in Figure 4.1, OJS interacts as well with ORKG to allow the use of access to the machine-actionable form of the selected paper. This integration can be performed through the adding of a new galley.

Chapter 5

Application

This chapter presents the application of the proposed method on a chosen article and describes the results of the implementation. We will start by setting up the realization of rewriting the use case paper "Gentsch" in a dynamic manner, then we will cite some results of our integration between the platforms.

5.1 Converting the paper to a reproducible form

In order to test our work, we firstly should select a scientific paper and take it as a test case article. Therefore, we decided to choose the article with the title "*Catch crop diversity increases rhizosphere carbon input and soil microbial biomass*" [101] and integrated it in all the steps of our work. Additionally, the code and the dataset related to the first figure of our selected paper code are kindly received from the author Dr. Gentsch in direct communication.

5.1.1 Reproducibility with Stencila

Preparation and setting up

After creating a free user account in Stencila hub, we will be able to create a project where we plan to host our data. The creation of a new project can be performed via the button *+ New Project* in the main dashboard of the user and will redirect the user to the interface displayed in the Figure 5.1. According to this figure, the establishment of a new project is too easy. Basically, the logged user needs only to choose a name for his/her project and then click on the button *+ Create Project*. The project can be either public or private. Once the project is created, the user

will be allowed to create a new file such as Google doc or sheet or to upload his/her code and data into this project through the interface described in the last chapter Figure 4.3. In our case, we will upload only two files to our project which are:

- R Markdown file: it will contain some parts of the used paper [101] and the R code to compute it.
- CSV File: it will be the dataset used for the computation.

New project

Account

Account that the project belongs to.

Name

Name of the project. Lowercase only and unique for the account. Will be used in URLs e.g. <https://hub.stenci.la/awesome-org/great-project>.

Public

Should the project be publically visible?

Figure 5.1: The creation of a new project interface in Stencila.

Realization

For rewriting the Gentsch paper in R Markdown, we have used the R Studio tool [61] to facilitate writing of the script. In order to avoid redundancy, we decided to not rewrite the complete article and select only the essential data such as title, authors, publication data, etc.

```
title: >-
```

```
  Catch crop diversity increases rhizosphere carbon input and soil  
  microbial biomass
```

```

isPartOf:
  volumeNumber: '56'
  isPartOf:
    title: Biology and Fertility of Soils
    identifiers:
      - name: doi
        propertyID: 'https://registry.identifiers.org/registry/doi'
        value: https://doi.org/10.1007/s00374-020-01475-8
        type: PropertyValue
    publisher:
      name: 'Springer Science and Business Media {LLC}'
      type: Organization
    type: Periodical
  type: PublicationVolume
---

```

The R Markdown code above represents a part of our script. It describes the elements of the paper in the form of objects. Essentially, the paper has a title and volume number. The volume has as well many properties such as title, identifiers and publisher which encapsulate also other properties like a name and value. For example, the object *isPartOf* has in our case three properties which are *volumeNumber*, *isPartOf* and *type*. The *volumeNumber* has **56** as value. Once we are done with the paper information, we have generated the Figure 5.22 that describes the Net ecosystem exchange of C between catch corp treatments. Therefore, we received the R code from Dr. Gentsch, and we included it in our R markdown file in the form of a chunk as follows.

```

chunk: Figure 1
:::
## Net ecosystem exchange (NEE) of C between catch crop treatments. Bars
  represent means SE; lowercase letters denote significant differences
  (p < `r p`) between treatments

```r
COL <- c("Fallow" = "slategray", "Mustard" = "red3" , "Mix4" = "orchid3",
 "Mix12"= "orange4")
SHP <- c("Fallow"=21,"Mustard"=22,"Mix4"=23, "Mix12"=24)

data <- read.csv2("data.csv", as.is=T)
data$cc_variant <- factor(data$cc_variant, levels = c("Fallow",

```

## 5.1. Converting the paper to a reproducible form

---

```
 "Mustard", "Mix4", "Mix12"))
data$NEE <- as.numeric(data$NEE)

ggplot(data, aes(x= cc_variant, y=NEE, fill= cc_variant))+
 geom_boxplot()+
 scale_fill_manual(values = COL, guide=FALSE)+
 geom_text(data= merge(df_NEE,Pos) ,
 aes(y=NEE-10,x=cc_variant, label=.group))+
 labs(x="Catch crop variant", y=expression("NEE (mg CO"[2]~"-
 C"~m^{-2}~h^{-1}~")"), fill="")+
 theme_myBW+scale_x_discrete(limits=c("Fallow", "Mustard", "Mix4",
 "Mix12"))
...
:::
```

---

The code above represents only a part of the Gentsch code. We can remark that it starts and closes with the ">:::" expression. Thereafter, the caption is stated and then the R code enclosed between these two expressions: ““r and ““. As outlined in the R code, we call the data from a CSV file and then use it to build the plot.

In the caption, we can notice that the value of  $p$  is represented in the form of R code to allow the users to compute it. Indeed, this p-value is calculated after computing the maximum p-value of all the p-values described in a table of combinations. The code that permit the generation of this table is developed as follows.

---

```
summary table
sum.lm <- glht(lm_NEE, linfct = mcp(cc_variant = "Tukey"))
#summary(sum.lm)$test$pvalue
glht.table <- function(x)
{
 pq <- summary(x)$test
 mtests <- cbind(pq$coefficients, pq$sigma, pq$tstat, pq$pvalues)
 colnames(mtests) <- c("Estimate", "Std Error", "z value", "p value")
 return(mtests)
}

df.summary <- data.frame(glht.table(sum.lm))
abc <- subset (df.summary, p.value<0.01)
maxValue <- max(abc$p.value)
p <- round(maxValue + 5*10^(-3), 2)
```

---

Following the same approach used to compute the figure, we will offer to the user the possibility to display or hide the table data based on this R code:

```
chunk:
:::
```r
df.summary$Combination <- row.names(df.summary)
df.summary2 <- df.summary[, c(5, 1, 2, 3, 4)]
df.summary2
```
:::
```

Since we completed the implementation of our code, it is only remains to upload it with its related dataset to the Stencila Hub. Furthermore, we should create a snapshot of the entire project in order to generate the HTML file of the paper.

## Results

The screenshot shows the Stencila interface with the following elements:

- (7)** Run Document button.
- (1)** Title: **Catch crop diversity increases rhizosphere carbon input and soil microbial biomass**
- (2)** Authors: Norman Gentsch, Jens Boy, Juan Daniel Kennedy Batalla, Diana Heuermann, Nicolaus von Wirén, Dörte Schweneker, Ulf Feuerstein, Jonas Groß, Bernahrd Bauer, Barbara Reinhold-Hurek, Thomas Hurek, Fabricio Camacho Céspedes, Georg Guggenberger
- (3)** Published: 2020-05-23
- (4)** Abstract, Introduction, Material and methods, Results. The Results section contains the text: **PLANT BIOMASS AND NET ECOSYSTEM EXCHANGE**  
The NEE decreased significantly with increasing catch crop diversity (Fig.1), suggesting increasing  $CO_2$ -C uptake from the atmosphere.
- (5)** Figure 1 placeholder with a play button icon. Below the placeholder, it says "No output to show".

At the bottom of the figure placeholder, there is a legend: "Net ecosystem exchange (NEE) of C between catch crop treatments. Bars represent means  $\pm$  SE; lowercase letters denote significant differences ( $p < 0.05$ ) between treatments".

Figure 5.2: The Gentsch paper output in Stencila.

## 5.1. Converting the paper to a reproducible form

---

Once the HTML file is created, we can display the output of the paper by clicking on the *Run* button. The Gentsch document will be shown in Figure 5.2. This figure shows an output of the Gentsch paper on Stencila. This output highlights the title of the paper (1), its authors (2), its publication date (3) and some part of its content (4). In addition to these data, the output of the paper allow the users to display and compute the plot via the buttons in (5). Indeed, the "eye icon" button permits the visualization and the hide of the related code. However, the "eye icon" enables the users to run the code and then generate the plot as an output. As well as figure, the p-value and the table (6) can be also executed when the user click on the button. All the executable instance of this paper can be run sequentially in one click via the button *Run Document* (7). Once the user executes the code related to the figure, the plot will be generated as shown in Figure 5.3.

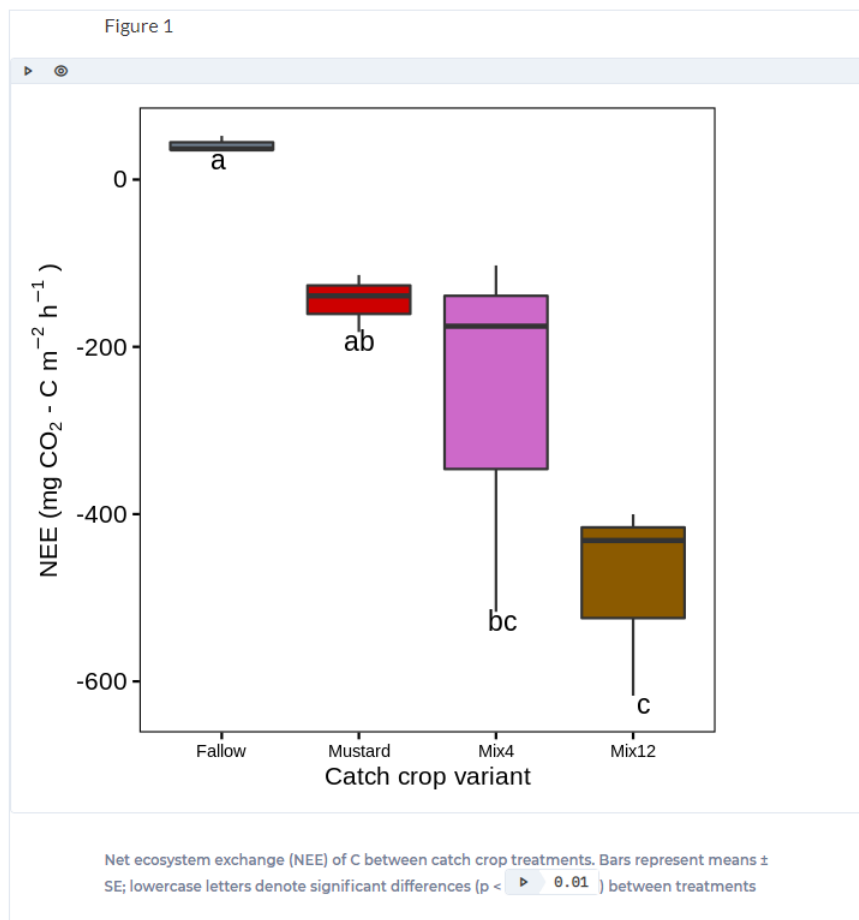
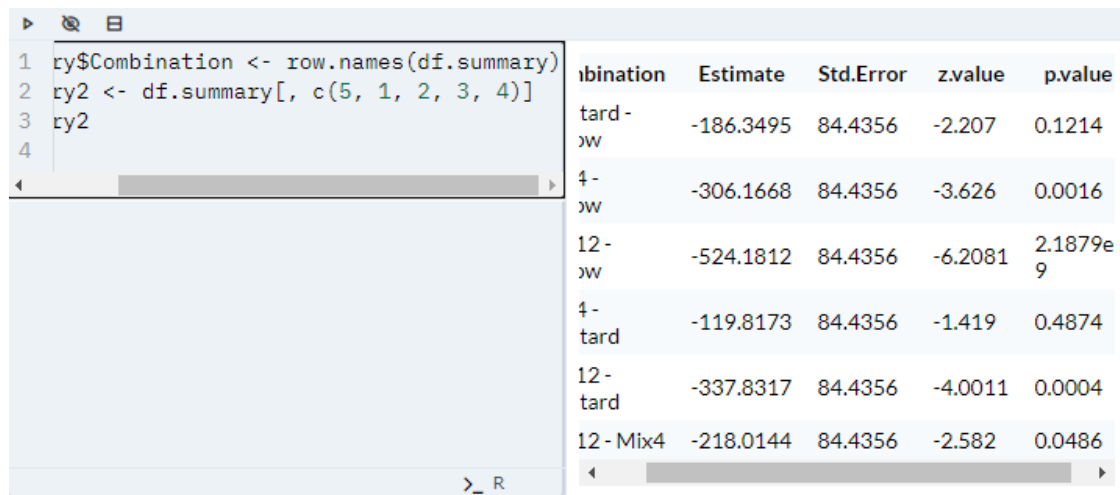


Figure 5.3: Output of the plot after running it in Stencila.

Figure 5.3 demonstrates also the computation of the p-value. Basically, the task of the execution of the p-value can not be performed before the execution of the plot code. Indeed, the p-value script requires the computed data from the plot code. Similarly, the table code is based also on the executed data of the last two scripts. The output of the table will be displayed as follows.



| Combination   | Estimate  | Std.Error | z.value | p.value      |
|---------------|-----------|-----------|---------|--------------|
| tard -<br>mix | -186.3495 | 84.4356   | -2.207  | 0.1214       |
| tard -<br>mix | -306.1668 | 84.4356   | -3.626  | 0.0016       |
| tard -<br>mix | -524.1812 | 84.4356   | -6.2081 | 2.1879e<br>9 |
| tard -<br>mix | -119.8173 | 84.4356   | -1.419  | 0.4874       |
| tard -<br>mix | -337.8317 | 84.4356   | -4.0011 | 0.0004       |
| tard - Mix4   | -218.0144 | 84.4356   | -2.582  | 0.0486       |

Figure 5.4: Output of the p-values table after running it in Stencila.

Figure 5.4 confirms the features of code visualization. Every user can display the code and its output together in one frame.

## 5.1.2 Reproducibility with Jupyter Book

### Preparation and setting up

As well as in Stencila, we prepared the suitable environment for Jupyter Book. Actually, in order to build our book, we need a file that contains the table of contents ("*toc.yaml*"), a configuration file ("*config.yaml*") and a Jupyter notebooks file ("*article.ipynb*").

Before continuing with this process, we should install the Jupyter Book python library. This library enables the creation, build, upgrade, and the control of our Jupyter Book. The installation can be performed through the package installer for python *pip* as follows.

```
pip install jupyter-book
```



## 5.1. Converting the paper to a reproducible form

For rewriting the Gentsch paper into Jupyter Notebook, we installed *Anaconda navigator* [100] as a desktop graphical user interface (GUI) included in Anaconda distribution. R package should be also added into this environment in order to enable Anaconda to compute the R code of the used paper. Indeed, other developing environments that support the building of Jupyter Books can be as well used. We can use, for example, Pycharm [102], Spyder [103], Atom [104], etc.

### Realization

We will input our paper information in the form of cells through the Anaconda Jupyter Notebook GUI. All the static data such as title, authors, publication information could be inserted in one Markdown cell as shown in Figure 5.5. Basically, we can use more than one Markdown cell in a way that each section of the paper has its own cell. However, it will be better to regroup all the text in one cell to optimize the execution and avoid an unorganized description of the data.

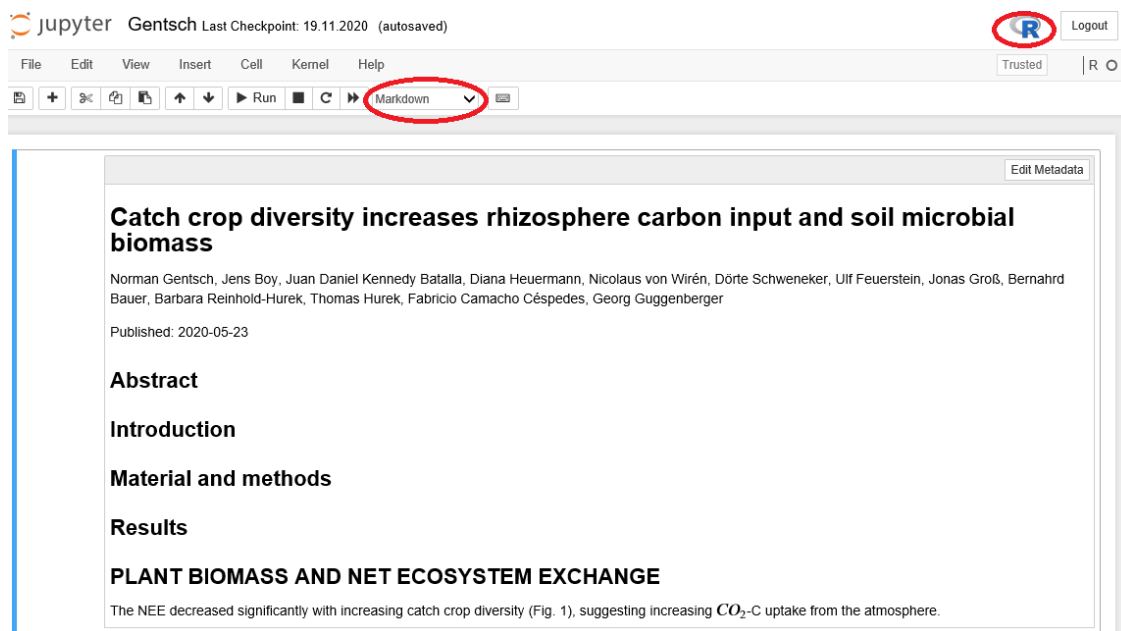
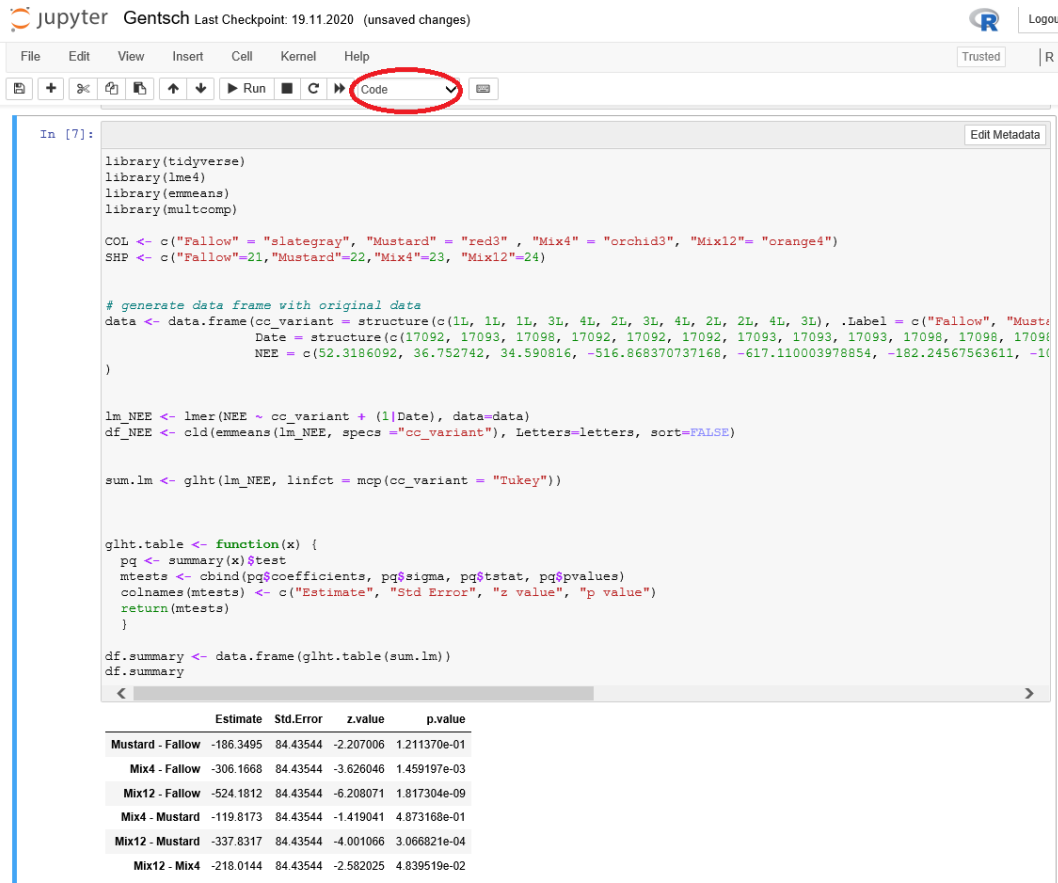


Figure 5.5: Gentsch’s paper markdown output in Anaconda Jupyter Notebook GUI

The figure above demonstrates the type of the cell (“markdown”) and the using of R as a kernel to compute the code. In fact, the user has the possibility to change the type of the selected cell via the displayed DropDown menu. However, the kernel of the Book is related to the programming language adopted in the code of the cells.

For example, Figure 5.6 uses a code cell to execute the R code responsible for the visualization of the displayed p-values table.



```

In [7]:
library(tidyverse)
library(lme4)
library(emmeans)
library(multcomp)

COL <- c("Fallow" = "slategray", "Mustard" = "red3", "Mix4" = "orchid3", "Mix12" = "orange4")
SHP <- c("Fallow"=21,"Mustard"=22,"Mix4"=23, "Mix12"=24)

generate data frame with original data
data <- data.frame(cc_variant = structure(c(1L, 1L, 1L, 3L, 4L, 2L, 3L, 4L, 2L, 2L, 4L, 3L), .Label = c("Fallow", "Mustard", "Mix4", "Mix12")),
 Date = structure(c(17092, 17093, 17098, 17092, 17092, 17092, 17093, 17093, 17098, 17098, 17098, 17098), .Label = c("17092", "17093", "17098")),
 NEE = c(52.3186092, 36.752742, 34.590816, -516.868370737168, -617.110003978854, -182.24567563611, -100.123456789012, -123.456789012345, -456.789012345678, -789.012345678901, -1012.345678901234, -1234.567890123456))

lm_NEE <- lmer(NEE ~ cc_variant + (1|Date), data=data)
df_NEE <- cld(emmeans(lm_NEE, specs = "cc_variant"), Letters=letters, sort=FALSE)

sum.lm <- glht(lm_NEE, linfct = mcp(cc_variant = "Tukey"))

glht.table <- function(x) {
 pq <- summary(x)$test
 mttests <- cbind(pq$coefficients, pq$sigma, pq$ststat, pq$pvalues)
 colnames(mttests) <- c("Estimate", "Std Error", "z value", "p value")
 return(mttests)
}

df.summary <- data.frame(glht.table(sum.lm))
df.summary

```

|                  | Estimate  | Std.Error | z.value   | p.value      |
|------------------|-----------|-----------|-----------|--------------|
| Mustard - Fallow | -186.3495 | 84.43544  | -2.207006 | 1.211370e-01 |
| Mix4 - Fallow    | -306.1668 | 84.43544  | -3.626046 | 1.459197e-03 |
| Mix12 - Fallow   | -524.1812 | 84.43544  | -6.208071 | 1.817304e-09 |
| Mix4 - Mustard   | -119.8173 | 84.43544  | -1.419041 | 4.873168e-01 |
| Mix12 - Mustard  | -337.8317 | 84.43544  | -4.001066 | 3.066821e-04 |
| Mix12 - Mix4     | -218.0144 | 84.43544  | -2.582025 | 4.839519e-02 |

Figure 5.6: Output of p-values table generated from the computation of code's cell.

The visualization of the R code in Figure 5.6 can be explicitly or optional performed. That's mean that the user has the choice either to show always the code with the table or to integrate a "show/hide" button as Stencila offers. Therefore, the metadata of the concerned cell should be updated by adding the tag *hide-input* as follows.

```

{
 "tags": [
 "hide-input",
]
}

```

Then the Jupyter Book will hide the cell but display the outputs as shown in Figure 5.10.

Once our paper is rewritten and our Jupyter Book library is installed, we can build our book based on these three steps:

1. Creating the book template: It will be performed via this command :

```
jupyter-book create Gentsch
```

This created book *Gentsch* can be customized and added to our table of contents, the configuration file, etc. After executing the command, we will get a new directory in our used path named "Gentsch". In this directory, we will have the configuration file "*\_config.yml*" which controls the management of Jupyter Book and offers us the possibility to set up metadata for the book such, title, authors, DOI, and even activate interactive buttons such as a Binder one to interact with any notebook via BinderHub (open framework for open research). In our example, we updated the "*\_config.yml*" file to add the URL of the repository on GitHub and the interactive buttons. Basically, we need to put the title of the Gentsch paper, its authors and a logo image. Additionally, the information related to the repository such as the URL, the path to the book and the branch should be updated. For the launch buttons, we used only two buttons, the *Binder* button and the *Live code* button.

---

```
Book settings
title: Catch crop diversity
author: Norman Gentsch
logo: logo.png

latex:
 latex_documents:
 targetname: book.tex

repository:
 url : https://github.com/AnouarGanfoud/Gentsch
 path_to_book : MA-jupyter
 branch : master #Branch of the repository

launch_buttons:
 thebe : true
 binderhub_url: "https://mybinder.org"
```

---

2. Generating the HTML file of the book: After creating our book, we will be able to convert each page into HTML thanks to the *jupyter-book* library. Indeed, this library permit the conversion of any ".ipynb", ".md" files into HTML that can be understood by a website. Therefore, we should run this command:

```
jupyter-book build Gentsch/
```

After executing the command above, a new directory called "\_build" will be created. This folder will contain all the used markdown and notebook files in HTML format as shown below.

| Name           | Änderungsdatum   | Typ             | Größe |
|----------------|------------------|-----------------|-------|
| _images        | 11.11.2020 23:01 | Dateiordner     |       |
| _sources       | 11.11.2020 00:12 | Dateiordner     |       |
| _static        | 11.11.2020 00:12 | Dateiordner     |       |
| .buildinfo     | 23.11.2020 21:20 | BUILDINFO-Datei | 1 KB  |
| genindex.html  | 23.11.2020 21:20 | HTML-Dokument   | 5 KB  |
| Gentsch.html   | 23.11.2020 21:20 | HTML-Dokument   | 31 KB |
| index.html     | 11.11.2020 00:12 | HTML-Dokument   | 1 KB  |
| objects.inv    | 23.11.2020 21:20 | INV-Datei       | 1 KB  |
| search.html    | 23.11.2020 21:20 | HTML-Dokument   | 6 KB  |
| searchindex.js | 23.11.2020 21:20 | JavaScriptdatei | 5 KB  |

Figure 5.7: HTML files in "\_build" folder generated by Jupyter Book.

3. Publishing the book online: Once we have our HTML files for each page, it is only remains to fuse them all together as a standalone HTML file that can be hosted online. Thus, we cloned our online repository on GitHub, and then we copied all of our book files and folders into this newly cloned repository. Thereafter, we establish a synchronization between the local and the remote repositories to push the changes online.

## Results

Once the GitHub pages feature is activated successfully in our repository, We will be able to see the whole project in a book format as shown below.

## 5.1. Converting the paper to a reproducible form

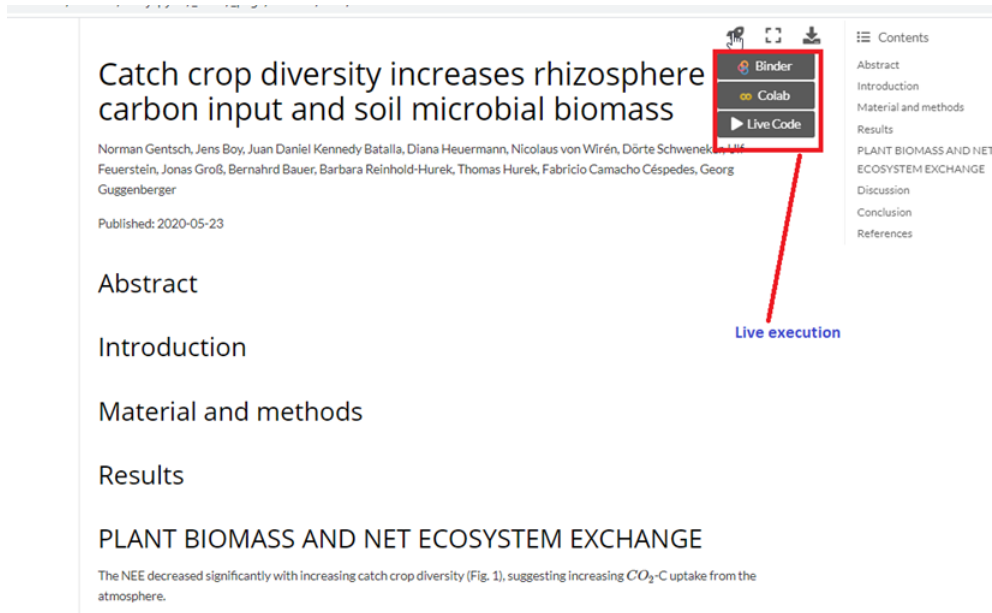


Figure 5.8: Output of the Gentsch paper with Jupyter Book.

According to Figure 5.8, we can start our live execution by choosing one of the platforms that is able to host the computation of our code. In our case, we offer the execution on Binder, Colab and Thebe. For example, the execution via Binder will redirect us to the following URL.



Figure 5.9: Computation of the Gentsch paper on Binder.

As we mentioned before, we can display the code explicitly with its related output, or we can give the user the possibility to show or hide the content of the code as shown in Figure 5.10.

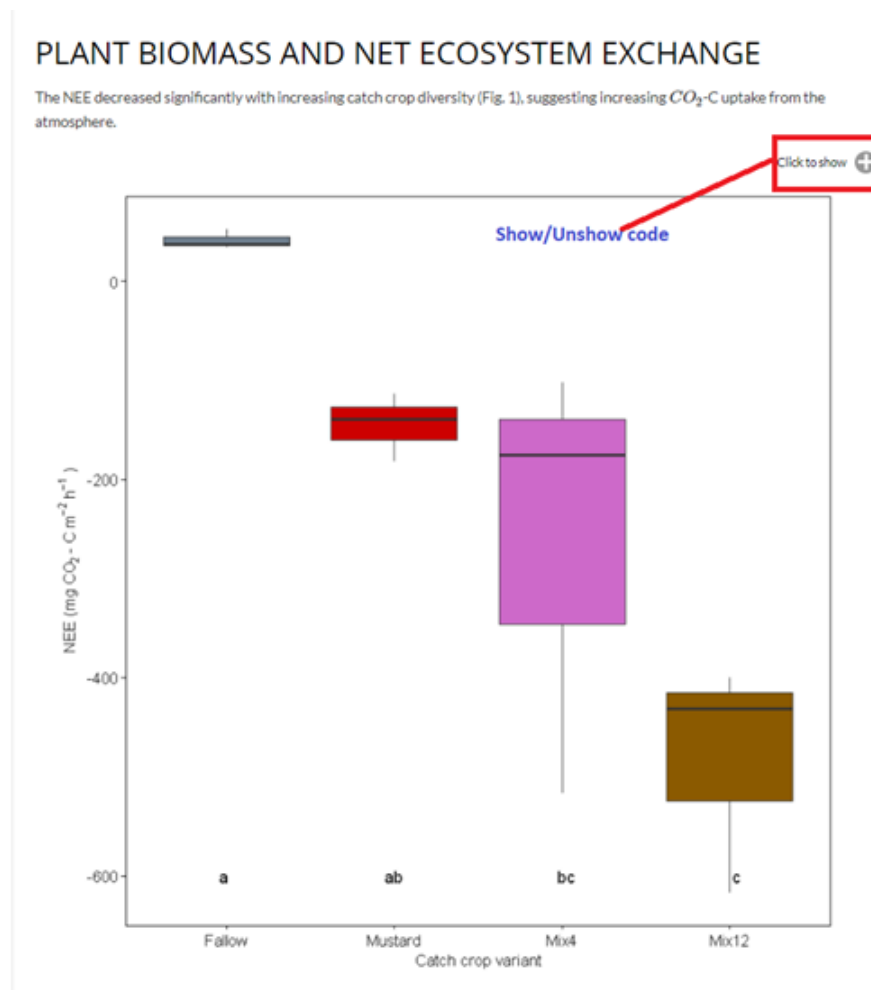


Figure 5.10: Output of the Gentsch plot with Jupyter Book.

## 5.2 Integration

### 5.2.1 Integration between ORKG and Stencila

#### Preparation and setting up

In order to improve the performance of the system and avoid the future bugs, we decide to install the platforms on our local machine. We will start with hosting the Stencila hub. Therefore, we need firstly to download the stencila/hub repository from GitHub, then install at least one of the required services such as Python ( $\geq 3.7$  with

## 5.2. Integration

---

pip and venv packages), Node.js (version  $\geq 12$  and NPM), Docker Engine, Docker Compose (used to run the service integration tests), Minikube or Kompose.

Once we have installed the required tools of Stencila, we should run each of the following commands in separate terminal windows:

---

```
Window 1
- make -C manager venv
- make -C manager create-devdb-sqlite
- make -C manager migrations
- make -C manager run

Window 2
- make -C broker run

Window 3
- make -C overseer run

Window 4
- make -C worker run
```

---

As well as Stencila, ORKG should be hosted. Basically, the installation of ORKG is performed in two steps:

1. Installation of the Backend API: We need firstly to download the Backend repository from GitHub. This repository consist of a Backend API for the ORKG based on the Java Spring Framework. It is developed in Kotlin as a proof-of-concept and for evaluating with several alternative architecture and technologies. This Backend API can be executed standalone. However, it requires many other services to start correctly such as Docker Engine, Docker Compose, Neo4j and PostgreSQL.

After installing these services, we should run the Docker Compose by typing

```
docker-compose up -d
```

Since we run and build the docker image, it only remains to start the API via this command:

```
./gradlew bootRun
```

This will start a server running on `http://localhost:8080`.

2. Installation of the Frontend: Once we have installed the Backend API of ORKG, we can begin with the installation of its Frontend. Therefore, we need also to clone the repository on our local machine. Thereafter, it will be necessary if we check that *Node.js* is installed (version  $\geq 10$ ). Then, we should go to the Frontend repository and install the dependencies via this command:

```
npm install
```

It is also required to copy the environment file **default.env** to **.env** as following.

```
cp default.env .env
```

Once we have completed the installation and the configuration of all the dependencies, we can run our ORKG Frontend by typing:

```
npm run start
```

The ORKG local instance will be available through this URL:

```
http://localhost:3000/
```

The installation of the ORKG package in the development tool is required. In our case, we used *PyCharm* [102] and installed the suitable ORKG package as demonstrated below.

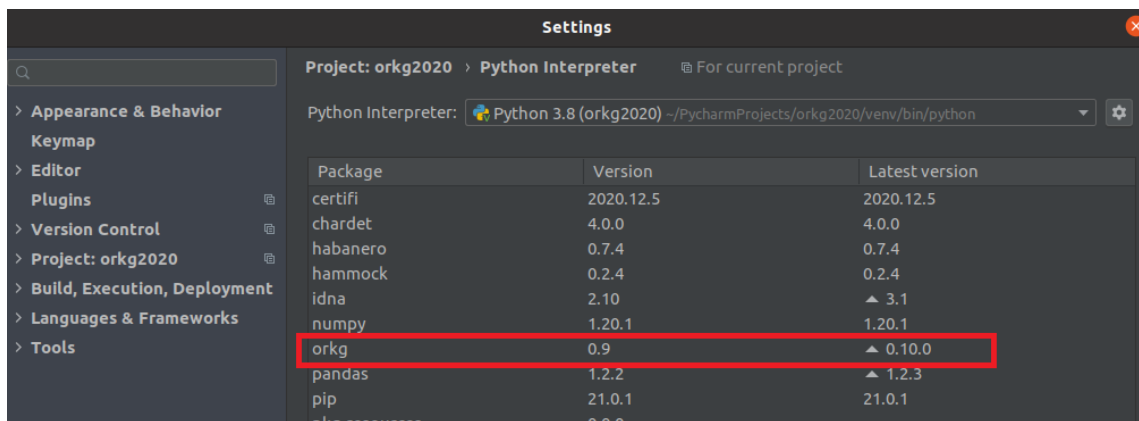


Figure 5.11: ORKG package in PyCharm.



### Realization

After installing the two platforms, we will try to simulate how ORKG can harvest data from Stencila projects. Therefore, we will firstly start with creating a user account in ORKG and establish the connection with it as following.

```
from orkg import ORKG
orkg = ORKG(host="http://localhost:8080",

 creds=('my email address', 'password'))
```

Indeed, we import the base class from the ORKG package, then we logged in using our account credentials. (The credentials given above are only an example). Since we are connected to the system, we can import our CSV dataset into ORKG. The easiest way to do that is using the python library as following.

```
datasetID = orkg.resources.save_dataset(
 file="/home/anouar/Downloads/pvalue.CSV",
 label="Summary of p-values",
 dimensions=["Combination", "Estimate",
 "Std.Error", "z.value", "p.value"])
```

Based on the command below, we can identify the path of the CSV file in our machine, the label and the dimensions which represent the columns of the dataset. After saving the dataset, we need to determine its ID by using this command:

```
print(datasetID.content)
```

Once we have the ORKG resource ID for the dataset, we will start creating a machine-readable description of the statistical hypothesis tests. The previous resource ID will be there needed as an object of the inputted dataset. Therefore, we will use the ORKG python lib native format JSON with the including of some ORKG terms such resource and predicate IDs:

---

```
paper = {
 "paper":{
 "title":"Catch crop diversity increases rhizosphere carbon input and
 soil microbial biomass",
 "doi":"10.1007/s00374-020-01475-8",
 "authors":[
 {
 "label":"Norman Gentsch",
```

---

```

 "orcid": "0000-0003-1166-8973"
 },
 {
 "label": "Juan Daniel Kennedy Batalla"
 },
 "publicationYear": 2020,
 "publishedIn": "Biology and Fertility of Soils",
 "researchField": "R153",
 }
}

```

---

The JSON code above create a paper object which includes some JSON keys and its values to describe the Gentsch paper. In order to avoid unnecessary information, we put it only a part of the JSON code involving title, DOI, some authors, research field and some publication data. We can notice that most of the values are inputted as String that contains the information or the ID. The output of this code snippet is shown in Figure 5.12.

Furthermore, the integration between ORKG and Stencila will be only guaranteed when we include the computed data that we used in Stencila into ORKG. Therefore, we need to add a contribution to the Gentsch paper. This contribution has the dataset as material, the adopted R function to generate the plot as approach, the computed p-value and the summarized table as a result. The code below shows the declaration of the added contribution with the name *Contribution 1*:

---

```

"contributions": [
 {
 "name": "Contribution 1",
 "values": {
 "P32": [
 {
 "label": "Estimation of the statistical difference in
 NEE of C between catch crop treatments",
 "class": "Problem",
 "values": {}
 }
]
 }
 }
]

```

---

This contribution has as a value the key *P32* which refers to the ID of the property *has research problem*. This property contains the label of the research problem discussed in the Gentsch paper, the class and a list of values. In addition to the research

problem, *Contribution 1* encapsulates other values such as the property "*utilizes*", described as following.

---

```
"P5047": [
 {
 "@id": "R70818"
 }
]
```

---

*Utilizes* is referred in ORKG via the ID *R70818* which represents the resource ID of the dataset imported via the CSV file.

The regression function used by Dr. Gentsch to compute the plot displayed in Figure 5.22 is described in JSON as follows.

---

```
"P2": [
 {
 "text": "Regression",
 "datatype": "xsd:string"
 }
]
```

---

Basically, the *P2* key is related to the property *employs* in ORKG.

Finally, we have to outline the result of this contribution. Therefore, we used the key *P1* referred to the ORKG property *yields*:

---

```
"P1": [
 {
 "label": "Maximum significant p-value",
 "values":
 {
 "P9004": [
 {
 "text": "0.0018",
 "datatype": "xsd:string"
 }
]
 }
 },
 {
 "@id": "R70800"
 }
]
```

---

The *yields* key above contains two values or two results: the computed p-value and the table that summarizes all the p-values of the different combinations. This table will be added in the form of a dataset imported from a CSV data. This dataset has the resource ID *R70800*. However, the p-value computed after calculated the maximum between all the possible p-values has the label *Maximum significant p-value*. Indeed, this p-value will be encapsulated in the other property with the *P9004* named *Specified numeric value*. The object of this property is the actual number **0.018**. After completing the writing of the JSON description, we can create a Python script that adds the paper to our ORKG instance as follows.

```
orkg.papers.add(paper)
```

## Results

Once the Gentsch paper is added to the ORKG instance, we can open it either by giving its ID in the URL or by using the search feature in ORKG. The output of our article information will be shown as following.

The screenshot displays the ORKG interface for the paper "Catch crop diversity increases rhizosphere carbon input and soil microbial biomass". The top section, labeled (1), shows the title, authors (Norman Gentsch, Jens Boy, Juan Daniel Kennedy Batalla, Diana Heuermann, Nicolaus von Wirén, Dörte Schwenecker, Ulf Feuerstein, Jonas Groß, Bernhard Bauer, Barbara Reinhold-Hurek, Thomas Hurek, Fabricio Camacho Céspedes, Georg Guggenberger), the year 2020, the journal Soil Science, and the DOI 10.1007/s00374-020-01475-8. The bottom section, labeled (2), shows the "Contribution 1" details, including the research problem "Estimation of the statistical difference in NEE of C between catch crop treatments", the contribution data table, and the yields table.

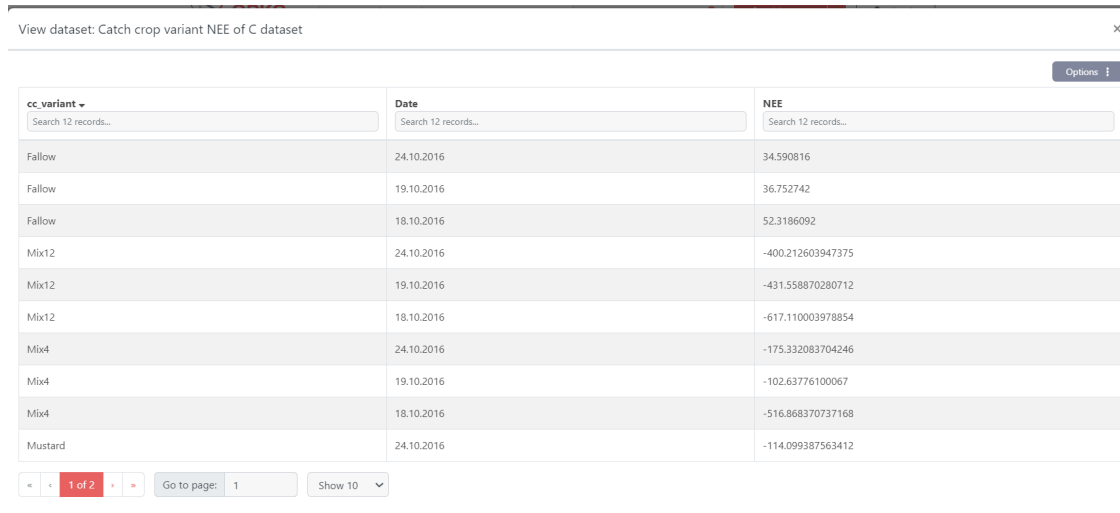
| Contribution 1                                                                     |                                            |
|------------------------------------------------------------------------------------|--------------------------------------------|
| <b>Research problems</b>                                                           | <input type="checkbox"/> Add to comparison |
| Estimation of the statistical difference in NEE of C between catch crop treatments |                                            |
| <b>Contribution data</b>                                                           |                                            |
| Employs                                                                            | Regression                                 |
| Utilizes                                                                           | Catch crop variant NEE of C dataset        |
| Yields                                                                             | Maximum significant p-value                |
|                                                                                    | Summary of p-values                        |

Figure 5.12: Gentsch’s paper presentation in ORKG.

The figure above shows the information related to our paper (1) such as title, authors, DOI and publication information.

## 5.2. Integration

Figure 5.12 displays as well the contribution data in the form of a grid. According to the JSON code, *Contribution 1* has three keys which are represented in three rows in the output. For the property *utilizes*, we can visualize the dataset in tabular form by clicking on the icon in front of the label.



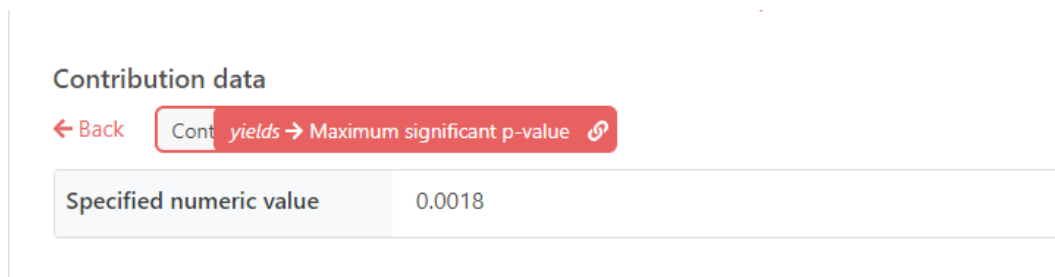
| cc_variant | Date       | NEE               |
|------------|------------|-------------------|
| Fallow     | 24.10.2016 | 34.590816         |
| Fallow     | 19.10.2016 | 36.752742         |
| Fallow     | 18.10.2016 | 52.3186092        |
| Mix12      | 24.10.2016 | -400.212603947375 |
| Mix12      | 19.10.2016 | -431.558870280712 |
| Mix12      | 18.10.2016 | -617.110003978854 |
| Mix4       | 24.10.2016 | -175.332083704246 |
| Mix4       | 19.10.2016 | -102.63776100067  |
| Mix4       | 18.10.2016 | -516.868370737168 |
| Mustard    | 24.10.2016 | -114.099387563412 |

Figure 5.13: Visualizing the dataset in the form of a tabular.

According to Figure 5.13, the data tabular is displayed in the form of a popup. ORKG offers the user the possibility to sort the data and search for a particular value.

For the property *yields*, we have two values:

- The table of all p-values represented in the form of a tabular.
- The computed p-value which calculated after selected the maximum significant value from the table. Basically, by clicking on this value, the exact value will be shown as follows.



**Contribution data**

← Back Cont yields → Maximum significant p-value

Specified numeric value 0.0018

Figure 5.14: Maximum significant p-value output in ORKG.

## 5.2.2 Integration between OJS and Stencila

### Preparation and setting up

The integration between the open journal systems and Stencila requires the installation of the two frameworks. After the hosting of Stencila, it only remains to install OJS on our local machine. Therefore, we need firstly to clone the OJS from GitHub, then install the Docker Engine and Docker Compose services. Once we have the OJS directory on our machine, we can open it and choose which version we want to install. In our case, we installed the version "3.2.0.1":

```
$ cd versions/3_2_0-1/alpine/apache/php73
```

Furthermore, we should start the stack using this command:

```
$ docker-compose up
```

Through this command, Docker compose will pull images from the docker Hub and do all the hard work to mount a functional OJS stack.

Furthermore, it is only still to access to the localhost URL "**http://127.0.0.1:8080**" and continue the installation process on the web interface. We need to check there that the database connection credentials are valid and match the following info:

- Database driver: `mysqli` (or "`mysql`" if `php <7.3`)
- Host: `db` (which represents the name of the container in the internal Docker network)
- Username: `ojs`
- Password: `ojsPwd`
- Database name: `ojs`
- We have to uncheck "Create new database" and "Beacon"
- Upload directory: "`/var/www/files`"

Now, we can use our OJS locally and create and publish submissions. However, our data is not persistent and all the work can be deleted when the docker is stopped. To avoid that, we should follow these three steps:

1. Stop the container via this command:

```
$ docker compose stop
```

2. Open the **”docker-compose.yml”** configuration file and uncomment the volume lines that we want to keep it
3. Restart the container through the following command:

```
$ docker compose up -d
```

In step 2, we need to be sure that our folders exist. So, to keep persistent database, uploads (public and private files) and our configurations, those will be the lines to uncomment:

---

```
24: volumes:
25: - ./volumes/db:/var/lib/mysql
...
43: volumes:
44: # - /etc/localtime:/etc/localtime
45: - ./volumes/private:/srv/files
46: - ./volumes/public:/var/www/html/public
47: # - ./volumes/logs/app:/var/log/apache2
48: - ./volumes/config/ojs.config.inc.php:/var/www/html/config.inc.php
```

---

We left some lines commented, but this doesn't affect the functionality of the hosted OJS version. We can as well edit these volumes in a flexible way to configure OJS. For example, it is possible to access and save the apache logs outside the container, or to create a new plugin by mapping this line *”/var/www/html/plugins/general/Theme”*. In general, these volumes can be considered as a door to the container and not only a way to ensure data will be persistent.

### Realization

Once we have completed the installation and configuration of OJS, we can start including the Gentsch paper as an online submission. Therefore, we need firstly to create an issue of this submission. This issue will encapsulate a collection of articles and should contain publication data such as a title, a volume identification and number as following.

**Issue Management: Vol. 56 No. 943–957 (2020): Biology and Fertility of Soils**
✕

Table of Contents
Issue Data
Issue Galleys
Help

**Date Published**

2021-03-14

**Identification**

56

943–957

2020

*Volume*
*Number*
*Year*

Biology and Fertility of Soils

*Title*

Volume
  Number
  Year
  Title

**Description**

📄
📁
**B**
*I*
U
🔗
🌀
<>
🔄
🖼️
📎

Figure 5.15: Issue data in OJS.

According to Figure 5.15, we can notice that the user can select, via the checkbox, which info is allocated to the issue. This issue can enclose one or more submissions. Indeed, the Gentsch paper will be inputted as a submission. By clicking on the button "New submission", we will start the creation process by determining the suitable policy as following.



## 5.2. Integration

**Submit an Article**

1. Start   2. Upload Submission   3. Enter Metadata   4. Confirmation   5. Next Steps

**Section Policy**  
Section default policy

**Submission Requirements**  
You must read and acknowledge that you've completed the requirements below before proceeding.

- The submission has not been previously published, nor is it before another journal for consideration (or an explanation has been provided in Comments to the Editor).
- The submission file is in OpenOffice, Microsoft Word, or RTF document file format.
- Where available, URLs for the references have been provided.
- The text is single-spaced; uses a 12-point font; employs italics, rather than underlining (except with URL addresses); and all illustrations, figures, and tables are placed within the text at the appropriate points, rather than at the end.
- The text adheres to the stylistic and bibliographic requirements outlined in the Author Guidelines.

**Comments for the Editor**

**Submit As \***  
Submit in any of the following roles if you would like to be able to edit and publish this submission yourself: Journal manager

- Journal manager
- Author

Yes, I agree to have my data collected and stored according to the [privacy statement](#).

Save and continue   Cancel

Figure 5.16: Creating a submission in OJS: Policy section.

Through the policy section, the submission requirements (1) have to be admitted. Basically, we have to agree with some concerns listed in the Figure 5.16 such as the type of the submission file and that the submission has not been previously published, etc. Additionally, we should specify our role in this journal by choosing between an author and a journal manager. In our case, we are a journal manager of the Gentsch paper, so we can select it as a role, then save and continue to be redirected to the next step of the creation process. During this step, the user can upload a submission file such as HTML file to use it as a layout or to skip the step and keep the default layout of OJS. For our example, we will use the HTML file generated from Stencila as layout and upload to the submission.

Furthermore, we will move to the third step of the submission process and define the metadata of our journal as shown in the following figure.

**Prefix**  **Title \***

*Examples: A, The*

**Subtitle**

**Abstract \***

Rich text editor toolbar: Bold, Italic, Underline, Bulleted List, Numbered List, Undo, Redo, Link, Unlink, Source, Image, Table, Print.

**List of Contributors** (3) [Add Contributor](#)

| Name                    | E-mail                   | Role            | Primary Contact                     | In Browse Lists                     |
|-------------------------|--------------------------|-----------------|-------------------------------------|-------------------------------------|
| ▶ anouar2002 anouar2002 | ganfoud.anouar@gmail.com | Journal manager | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

**Submission Metadata**

These specifications are based on the Dublin Core metadata set, an international standard used to describe journal content.

**Additional Refinements**

**Keywords**

Figure 5.17: Creating a submission in OJS: Defining the metadata.

As we did with the previous frameworks, we should attribute a title (1) to the OJS journal and some optional metadata such as prefix and subtitle. In contrary to Stencila and ORKG, OJS obliges the user to add an abstract (2) to describe briefly the idea and aim of the journal. The authors can be also added by clicking on the link *Add Contributor*. Indeed, a popup form that presents the contributor metadata such as name, ORCID, contact, etc. will be opened. After finishing the process of all the three previous steps, we will be able either to confirm our given metadata and publish the submission or to go back and modify the info.

Once we published the Gentsch article in the form of a submission, we can start to integrate it with the computed version of Stencila. Therefore, OJS offers a feature called Galley that permit the users to allocate the journal with another different versions like PDF, HTML, etc. In our case, we will include an HTML file as Galley.

## 5.2. Integration

---

This HTML file will encapsulate the Stencila output in the form of an HTML iframe as following.

---

```
<iframe
 src="https://anouar.stencila.io/gentsch2020/"
 frameborder="0"
 style="position: absolute; top: 80px; left: 0; width: 100%; height:
 100%"
 scrolling="yes"
 title="Reproducible Article"
>
</iframe>
```

---

Indeed, we should give the link of our Gentsch project in Stencila using the attribute "src" of the HTML element "iframe" as shown in the code above.

Adding a new galley in OJS requires that the journal isn't published. Therefore, we have to unpublish our Gentsch journal, then access to its publication tab (1) and select the section "Galleys" from the navigator sidebar as shown in Figure 5.18.

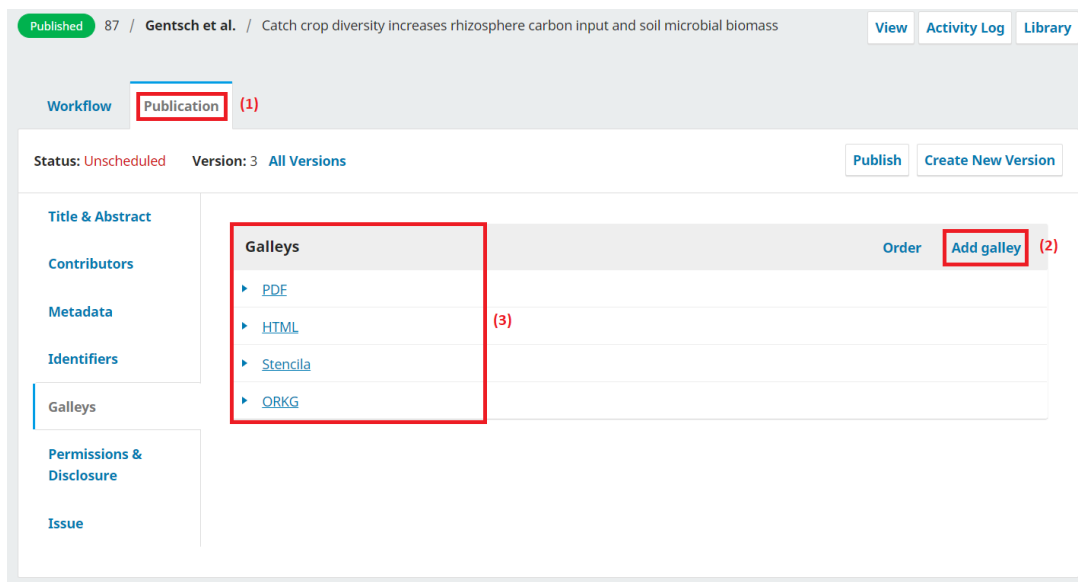


Figure 5.18: Adding a galley to a submission in OJS.

Figure 5.18 displays the list of the galleys (3) related to our paper. Actually, the user can add a new galley by clicking on the link *Add Galley* (2). This link will open a popup form that enables us to choose the label, path and language of the

galley. Thereafter, we can upload a file to this galley and select its article component (Article Text, Research Material, Dataset, etc.). Dependent files such images can be also added to the galley. In our case we will name our galley *Executable version* and allocate it only an HTML file that permit the visualization of the Stencila dynamic version. However, we shouldn't forget to change the Gentsch project settings in Stencila by allowing OJS to have access to the project.

## Results

After publishing our submission, we will be able to access to the output of our journal by clicking on *View website* link. This link will redirect us to the published output of our journal as displayed below.

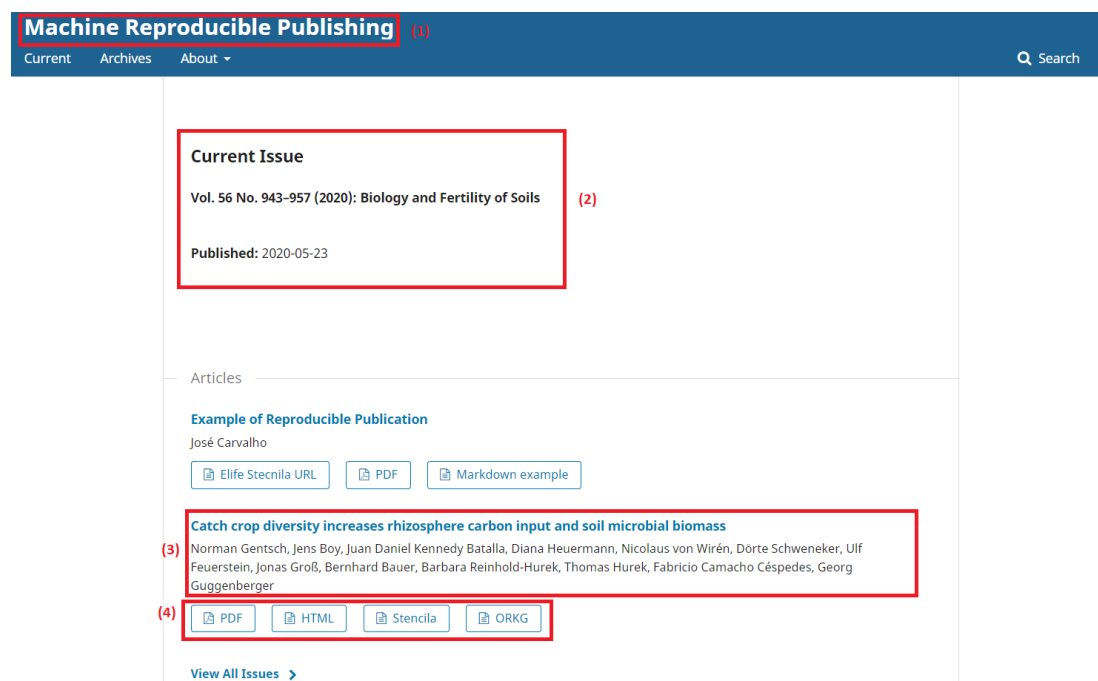


Figure 5.19: Index of the Gentsch journal in OJS.

According to Figure 5.19, we can notice that the index web page of the OJS journal contains four main elements which are:

- The title of the journal (1): we called Machine Reproducible Publishing in our case.
- Issue metadata (2): such as the volume number and the publication date.

## 5.2. Integration

- The list of articles included in this issue (3): For example, the title and authors.
- The list of the project galleys (4): In our case, we include several versions of the journal including PDF, HTML, Stencila and ORKG.

To access to the article content, we need only to click on the article title. Indeed, the user will be redirected to the default view of the article in OJS.

Home / Archives / Vol. 56 No. 943-957 (2020): Biology and Fertility of Soils / Articles

### Catch crop diversity increases rhizosphere carbon input and soil microbial biomass

**Norman Gentsch**  
<https://orcid.org/0000-0003-1166-8973>

**Jens Boy**

**Juan Daniel Kennedy Batalla**

**Diana Heuermann**

**Nicolaus von Wirén**

**Dörte Schwenecker**

**Ulf Feuerstein**

**Jonas Groß**

**Bernhard Bauer**

**Barbara Reinhold-Hurek**

**Thomas Hurek**

**Fabrizio Camacho Céspedes**

**Georg Guggenberger**

(4)

PDF HTML Stencila ORKG

Published  
2021-03-10 — Updated on 2021-03-10

Versions  
2021-03-10 (3)  
[2021-03-10 \(2\)](#)  
[2021-03-10 \(1\)](#)

Issue  
[Vol. 56 No. 943-957 \(2020\): Biology and Fertility of Soils](#)

Section  
Articles

f t e m + 0

DOI: <https://doi.org/10.1007/s00374-020-01475-8>

**Keywords:** Catch crops, Cover crops, Agrobiodiversity, 13C pulse labelling, Rhizosphere C-transfer, Soil microbiome

(2)


**Abstract**

Catch crops increase plant species richness in crop rotations, but are most often grown as pure stands. Here, we investigate...

(3)

Figure 5.20: Output of the Gentsch article in OJS.

Figure 5.20 describes the default view of our article. Indeed, we can identify a part of inputted metadata such as the contributors (1), the DOI and the keywords (2), and the abstract of the article (3). Furthermore, the galley of the journal (4) are also shown. The integration with the related dynamic version of Stencila can be performed through the *Stencila* galley. This galley will call the computed version of the article already build in Stencila as shown below.



**Machine Reproducible Publishing** (1)  
Current Archives About

▶ Run Document (2)

## Catch crop diversity increases rhizosphere carbon input and soil microbial biomass

Norman Gentsch, Jens Boy, Juan Daniel Kennedy Batalla, Diana Heuermann, Nicolaus von Wirén, Dörte Schwenecker, Ulf Feuerstein, Jonas Groß, Bernahrd Bauer, Barbara Reinhold-Hurek, Thomas Hurek, Fabricio Camacho Céspedes, Georg Guggenberger

Published: 2020-05-23

### Abstract

Catch crops increase plant species richness in crop rotations, but are most often grown as pure stands. Here, we investigate...

### Results

#### PLANT BIOMASS AND NET ECOSYSTEM EXCHANGE

The NEE decreased significantly with increasing catch crop diversity (Fig. 1), suggesting increasing  $CO_2$ -C uptake from the atmosphere.

Figure 1

▶ ©

*No output to show*

Net ecosystem exchange (NEE) of C between catch crop treatments. Bars represent means  $\pm$  SE; lowercase letters denote significant differences ( $p < 0.05$ ) between treatments

Figure 5.21: Output of the computed version of Stencila in OJS.

## 5.2. Integration

Although we switch to the article's reproducible form which is provided by Stencila, we can remark that the header of our OJS journal (1) still persists as shown in Figure 5.21. Additionally, the button *Run Document* (2) appears to ensure the executability of this version which permits the computation of plots and p-values. In this manner, we have guaranteed the integration between Stencila as a reproducible framework for the Gentsch paper and OJS as a static traditional framework.

Moreover, the *ORKG* galley displayed in Figure 5.20 permits the user to access to the machine-actionable version of the article. This version is namely described through ORKG as shown in Figure 5.22.

The screenshot displays the ORKG interface within an OJS journal. At the top, a blue header (1) contains the text "Machine Reproducible Publishing" and navigation links "Current", "Archives", and "About". Below this, the ORKG logo and navigation menu are visible. The main content area shows the article title "Catch crop diversity increases rhizosphere carbon input and soil microbial biomass" (2), the year "2020", and a list of authors including Norman Gentsch, Jens Boy, Juan Daniel Kennedy Batalla, Diana Heuermann, Nicolaus von Wirén, Dörte Schwenecker, Ulf Feuerstein, Jonas Groß, Bernhard Bauer, Barbara Reinhold-Hurek, Thomas Hurek, Fabricio Camacho Céspedes, and Georg Guggenberger. The article is published in "Biology and Fertility of Soils" with a DOI of 10.1007/s00374-020-01475-8. Below the metadata, a section titled "Contribution 1" (3) contains a "Research problems" field with the text "Estimation of the statistical difference in NEE of C between catch crop treatments" and an "Add to comparison" checkbox. The "Contribution data" section includes three rows: "Employs" with "Regression", "Utilizes" with "Catch crop variant NEE of C dataset", and "Yields" with "Maximum significant p-value" and "Summary of p-values".

Figure 5.22: Output of the machine-actionable version of ORKG in OJS.

According to Figure 5.22, we can observe the integration between the ORKG's machine-actionable form represented by the article metadata (2) and its contribution (3) and the OJS framework described through the header (1).

# Chapter 6

## Discussion

In this chapter, we discuss our work by listing the benefits, limitations and the challenges we faced. We also compare our work to other related works, and suggest some future research directions.

### 6.1 Advantages

We used several frameworks for publishing machine actionable reproducible scholarly knowledge. Indeed, we used Stencila and Jupyter Book as reproducible computing environments, ORKG as infrastructure for machine-actionable scholarly knowledge and OJS as traditional publication platform. We discovered Stencila as a new open source platform for creating executable documents. Indeed, we think that Stencila will have a leading role in scholarly communication and its infrastructure. It helped us in quickly understanding the concept of computationally reproducible papers. Additionally, using Stencila was very easy, and didn't require specific foreknowledge. However, since Stencila is a young and maturing system, we did occasionally rely on its community for help. Their small team was always extremely helpful. Thanks to Stencila, we have managed to provide a new dynamic version of our use case paper [101]. The first author was very satisfied with the results and suggested he may consider using this approach for a future paper before it is published.

In addition to Stencila, we tested Jupyter Book as an alternative solution for publishing a reproducible paper. In fact, Jupyter Book improved the reproducibility of our article by allowing us to establish the connection with different online services like Binder, Google Colab and Thebe.

Furthermore, the use of Stencila and Jupyter Book drove us to learn several additional technologies ranging from Python libraries and Jupyter Notebook to R



and the Markdown language.

With the integration of Stencila, ORKG and OJS we test how far can we go with the reproducibility of machine actionable scholarly knowledge. Indeed, involving ORKG in our work permits us to understand how can we gather data from other projects and publish it in a structured manner.

Moreover, the Stencila and OJS integration includes new developments by offering us the possibility to convert, execute and preview our executable research article offline. This integration allows us to build a package that makes it simpler and quicker for the scientists to preview the article in their local environment, before sending the completed project to Stencila Hub for future submission to a journal.

To summarize, this work helps us to discover new frameworks and technologies and to improve the reproducibility by ensuring transparency, verification and knowledge transfer among researchers [105].

## 6.2 Limitations and challenges

In the previous section of this thesis, we explained the advantages that we gained in adopting several knowledge frameworks to improve the publishing of machine actionable reproducible research as a main aspect in the conduction of scholarly knowledge. However, several issues arise when we decide to include reproducibility in our work in practice. As we mentioned before, Stencila represents a new framework and is rapidly but still in development. Indeed, many features were inoperable such as the cross-reference or the export of the figures in the local machine. Many Python and R libraries were not supported by Stencila such as *"pandas"* and *"multcomp"*, among others. Fortunately, all these issues are immediately fixed by the Stencila community after our request.

Besides, the Stencila Hub was not always stable due to the frequent updates, and sometimes we couldn't create a snapshot and upload the new changes. This drove us consider hosting the Stencila Hub. However, as described in the previous chapter, the installation was complicated, and we couldn't have a full functional version due to many missing dependencies in the GitHub repository.

In addition, we faced also some challenges during the installation of OJS and ORKG. Basically, many classes, properties and resources such as *"Research Field"* and *"Specified numeric value"* are not added during the hosting of ORKG. Therefore, we contacted the ORKG team and configured the required vocabulary manually through Python scripts.

There is no doubt that the installation of OJS was the easiest comparing to the other platforms. Nevertheless, there was a problem of data persistence and all our

saved work is always deleted when we restart the docker compose tool. This issue was fixed after receiving supports from the OJS community. Moreover, the integration between the OJS and Stencila was not effortless. The output of the dynamic version generated by Stencila in our OJS journal is blocked in every used web browser. Indeed, our test article is created locally and Stencila refuses every web browser to display a localhost page if another site has embedded it. Therefore, we should host our OJS on an online server, then we submit our journal there in order to enable to integrate with OJS.

Reproducible research is difficult to implement since scientists are typically reluctant to share their data and code. In many cases, a scientist does not find good conditions to check if their project is reproducible and ready for use for other researchers. Actually, some Journals have strict restrictions, such as the disabling of some features like the search and the index of available data. The researchers will then be blocked to reproduce the published materials [106].

Finally, we stress that even though completed reproducible science supports by the finding of bugs and misconduct, it does not assure the accuracy of the results of a study. However, it allows us in detecting potential errors or bad use of computational approaches.

## 6.3 Comparisons

### 6.3.1 Comparing Stencila with Jupyter Book

Stencila and Jupyter Book are two open source projects that aim to build research articles, computational notebooks, or sophisticated interactive documents. In our work, we focus more on testing the features of Stencila. Indeed, there exist only few works that focus on improving science reproducibility with Stencila. Therefore, we tried to test and evaluate this platform and compare it with Jupyter Book. Based on several criteria, we have decided to establish a comparison between the Stencila Hub and Jupyter Book as explained in Table 6.1. According to this table, we can conclude that Stencila represents the best option for beginners thanks to its use and simple configuration, and namely its reproducibility of the individual values in sentences (the p-value example of our work). Nevertheless, it will be recommended to adopt Jupyter Book when the researcher wants to integrate more graphic features to his/her document and try the interactivity of his/her data in different servers. Moreover, Stencila is a young project, and we cannot expect all features offered by Jupyter Book. However, Stencila is improving fast.

### 6.3. Comparisons

Criteria \ Framework	Stencila Hub	Jupyter Book
Version model	Open Source, Free (for limited features)	Open source, Free
Setup and configuration	Simple: through the web interface of the hub, and it doesn't require any other configuration	Complicated: The creation and the publication of the book is performed via the CLI or terminal. Further configurations could be done by editing "( <i>.config.yml</i> )".
Publishing the project	Simple: via the web GUI, and after the generation of an HTML file	Complicated: via CLI, it is performed in two steps: building the book (generation of the HTML locally), then publishing the book on a server.
Hosting	Possible. However, some features still missing in the local version	Possible
Supported technologies	Markdown, Python, R and Julia	Jupyter Notebook (Python, R, etc.), Markdown
Interactivity with other servers	Online with its own Hub	Possible: Binder Hub, Jupyter Lab, Google Colab and Thebe Lab
Velocity	Fast (<5 seconds to show the plot in our paper)	Slower than Stencila (depending on running server)
Features	Missing features like Interactivity, saving images, etc.	It offers more features than Stencila Hub.
Stability	Occasional instability	Stable
Integration with other project	Some online journals such eLife and PLOS are integrated	Used by many projects [107] such as MalariaGEN [108], TensorDiffEq [109], etc.
Documentation	Poor	Very large documentation with interactive examples

Table 6.1: Stencila Hub VS Jupyter Book.

## 6.3.2 Comparison with related work

### Comparing with Stencila-eLife Integration

During our work, we have tried to integrate our test case paper with Stencila. We adopted Open Journal Systems as a framework for submitting and publishing our test case article online. Actually, this integration is based on the collaboration project between Stencila and eLife Magazine. Stencila and eLife plan to bring reproducible research for more authors and reduce the barriers to the authoring and sharing of reproducible scholarly knowledge. After evaluating the article with the title "*Replication Study: Transcriptional amplification in tumor cells with elevated c-Myc*" [68] which represents the only reproducible article of eLife, we can conclude that the authors used an HTML iframe to integrate the computed Stencila output into the eLife journal. Indeed, we have also adopted the same technique in our integration. However, we didn't style our article using CSS style-sheet as they did, and we only copied the HTML file from Stencila and pasted it in OJS. We have focused more on the functionality than on design and style. Comparing the two works, we can notice that both of them offer the user to switch between the version using a button or link on the top of the page. Furthermore, eLife offers users the possibility to download the article in many formats such as executable, article as PDF and figures as PDF. In our work, we only provide authors the possibility to generate a PDF file from the article. The download of the executable version is not possible. Moreover, the source code of the paper is accessible in the eLife version. In fact, the user will be redirected to the snapshot page of the project in Stencila. Nevertheless, we avoid allowing authors access to this page in order to prevent misconduct of data because of the instability of the Stencila platform. Moreover, the required code for computing the data is already accessible in the journal, so it won't be necessary to share another code snippets for other researchers.

In terms of technologies, the two works employed almost the same technologies. The eLife-Stencila integration applied Markdown for creating the formatted text and R for computing the data of plots and tables. They include CSS also for styling the output of the article. Similarly, we employed as well Markdown and R for writing our article, but we didn't involve styling with CSS due to the time constraints.

We conclude that the two works have more similarities than differences. Basically, the integration between Stencila and OJS is based on the eLife-Stencila project, and we tried to imitate the experience in a different platform. We have partially succeeded to establish this approach in terms of functionality and efficiency. However, some work remains to be done, especially related to the design and the usability of the page.

### Comparison with the CAP project

Besides the eLife-Stencila integration which can be compared to our OJS-Stencila implementation, we can as well evaluate the integration between ORKG and Stencila to some related works such as the CERN Analysis Preservation (CAP) project.

Both, ORKG and CAP aim to improve reproducibility by harvesting data from other projects. The service offered by CAP permits various users to manipulate several entry points and functionalities in order to make the research process easier. Thus, CAP's users can submit their content through a submission form similar to the OJS form. Moreover, the input of journal metadata can also be automatically performed via a machine-readable description in the form of a flexible JSON schemata, identical to our integration between Stencila and ORKG. Figure 3.3 describes the form that contains the links to code, data and physics information. Comparing this interface to the ORKG contribution interface shown in Figure 5.12, we can notice many similarities in terms of data visualization and organization. Indeed, the researchers in ORKG can visualize the metadata related to their documents by clicking on the data resource link. However, CAP generates the data automatically after include it as described in the figure.

Furthermore, CAP allows authors to give the permissions to their submissions. In other words, they will be able to control the privacy of their publications via a specific form displayed in Figure 3.4. In our work, we have faced also the issue of privacy of submissions. Stencila affords users the opportunity to share their projects with other collaborators or to keep them private. Additionally, the submitter in OJS can invite users to view his/her submission. Basically, he/she can as well manage the roles of these users by allocating a list of permissions for each role. Regarding adopted technologies, CAP supports JSON Schema in order to allow the reuse of analysis data and materials in an automated concept. Besides, the authors of the CAP project plan to integrate Jupyter Notebook as a programming language targets to provide and facilitate modern research approaches like reproducibility of scholarly knowledge. In addition to JSON and Jupyter Notebook, we have adopted several technologies in our work. Indeed, the project Stencila-ORKG was based on JSON, R and Markdown. Otherwise, Jupyter Notebook was involved in the building of our test case paper via Jupyter Book. In contrast to our integration, CAP uses existing collaboration tools such as GitLab and GitHub to attach the inputted code, and then permits the authors to follow the repository changes of the work.

We conclude that the CAP project has many common points with our work, represented essentially in the implementation of a web interface for researchers in order to preserve their materials and reproduce it in the future. Hence, the two projects differs partially in the methodology but shares together the same purpose.

## 6.4 Future works

Future investigations are necessary to validate the conclusions that can be drawn from this study. For example, the integration between ORKG and Stencila drove us to think about including the visualization of code in the ORKG research contributions. This could be considered by the ORKG team in order to improve the description of research contributions traditionally defined in scholarly papers. In our case, it would be more descriptive if we could include the R and Python scripts used for the computation of the plot and the table to the contribution metadata. As a result, users would be able to see how the results are computed as well as how Stencila displays the code snippets of some data. Therefore, we will have a full harvesting of materials provided by ORKG that can help us to better describe our integration with other platforms like Stencila or Open Journal Systems. However, it remains unclear if it is possible to compute also the used code in ORKG. This will represent a big challenge for the ORKG team, but it can take this framework many steps ahead in the world of reproducibility.

Looking forward, further attempts could prove quite beneficial to this literature. Indeed, the integration between Stencila and Google Docs [110] give us the idea to think about another integration of ORKG with the two frameworks. We can replace the CSV file hosted in Stencila Hub, and permit the description of our project dataset with an online CSV file created via Google Docs and hosted in Google Drive. Furthermore, the integration between Stencila and Google Docs will allow us to execute and embed reproducible figures inside Google Docs. This computation could be later visualized by ORKG thanks to its modern description of data. This may constitute the objective of future work such as an integration between Stencila and  $\LaTeX$  environments. Imagine enabling citing executable code of different projects in our  $\LaTeX$  paper! The data and code in  $\LaTeX$  could be shown and hidden according to user requirements. Moreover, users would be able to execute code and visualize their plots and tables and offer the possibility to other users to run as well the data and view how is it computed. All these potential integrations between Stencila and previous discussed environments will launch a revolution in the scholarly knowledge area by affording modern scientific best-practices for reproducible science through allowing users of all technical levels to concentrate more on data analysis and on authoring manuscripts, instead of software engineering.

In addition, including OpenAI GPT-3 [111] features in reproducibility might prove an important area for future research. This technology can be defined as a third-generation, sophisticated language model that adopt deep learning methods to generate human-like papers [112]. The implication of this technology for computing

a paper text can be very beneficial for researchers and helps them, for example, to identify hate speech and classify text as sexist or racist [113]. In particular, the computation of hate texts uses the ETHOS dataset [114] which is based on comments located in YouTube and Reddit. The ETHOS YouTube data is accumulated via a particular platform called "*Hatebusters*" [115]. This platform gathers comments from YouTube and allocate a "hate" score to them based on a support vector machine process. The data related to Reddit is accumulated from the Public Reddit Data Repository [116]. Based on these mechanisms, we can conclude that GPT-3 is a machine learning technique that adopts deep learning to produce texts similar to texts written by humans. Nevertheless, researchers still frustrated with reproducing the results of a machine learning research. The fact that attaching a source code to go along with a research paper helps a lot in controlling the credibility of a machine learning approach and creating on top of it. However, this still not a requirement for machine learning conferences. Thus, a lot of students and scientists who use these papers struggle with reproducing their results. The integration between machine learning technique such as GPT-3 and a framework that supports reproducibility like Stencila may encourage some researchers to include reproducibility in their machine learning projects.

# Chapter 7

## Conclusions

This thesis suggests that researchers can publish their academic works in a structured and reproducible manner to ensure that they can repeat the same analysis multiple times with the same results, at any point in that process [117] with the best possible machine support, since scholarly knowledge is machine actionable.

We integrated various frameworks with complementary goals and functionalities. Specifically, we have used Stencila and Jupyter Book for providing reproducibility and repeatability of data, ORKG for describing scholarly knowledge in machine actionable form, and finally OJS for publishing articles in a traditional manner and linking the various digital artefacts (static, dynamic, and machine actionable expressions of scholarly work). The adoption of these frameworks permits us to make the use of scholarly knowledge easier for scientists.

The production of (parts of) an academic article in a dynamic manner was the first corner stone of this study. This has encouraged us to describe the considered paper in a structured manner by integrating the data computed in Stencila in ORKG. This description will facilitate the task of finding and comparing the paper. Furthermore, our study focuses on publishing online articles in a reproducible way. Therefore, OJS was employed as an online intelligent host for the article. This host offers readers the possibility to convert the article to a dynamic instance by calling all the required resources from Stencila.

In summary, the measures we have described in this thesis represent the base of efficient and feasible steps toward reconstructing machine actionability and reproducibility of scholarly knowledge. Most of them have demonstrated efficiency, and are good adopted, evaluated and improved. Actually, these approaches are not an exhaustive list, it exists many other original and advancing studies for making academic research techniques more adequate and reliable [118]. Providing a solution to



---

an issue does not maintain its validity, and can be changeable depending on cultural criteria and incentives can incite further behavioural changes that are hard to predict. Indeed, some approaches may be useless and even destructive to the efficiency and reliability of science, even if theoretically they seem reasonable.

Nowadays, the domain of meta-science is increasing very quickly, with more than 2,000 relevant academic publications collected every year [119]. The majority of this literature establishes the evaluation of our study and the description of alternative solutions. What was previously defined as assumed may be discussed, such as the integration between the frameworks; for example, the provided data and code to compute in an article may sometimes be not 100% correct and could be falsified by researchers in order to only show the desired results. Suggested solutions may also produce other challenges; for instance, as long as the reproducibility process represents an indication for increasing the trust level of scientific studies, there is always doubt about which research merit to be reproduced and what would be the best reproducibility strategies. Furthermore, a current simulation indicates that reproducibility alone may not be enough to avoid incorrect results [120].

Basically, these warnings don't represent an excuse for inactivity. Reproducible science processes are at the core of research and essential to the scientific approach. How perfect to accomplish exact and effective knowledge aggregation is a scientific question; the most adequate suggestions will be recognized by a mix of excellent hypothesizing and dumb luck, by repetitive evaluation of the efficiency of every modification, and by a winnowing of several opportunities to the largely determined few.

# Bibliography

- [1] James G. Leyburn. “On the Shoulders of Giants: A Shandean Postscript. By Robert K. Merton. Foreword by Catherine Drinker Bowen. New York: The Free Press, 1965. 290 pp. 5.95”. In: *Social Forces* 44.4 (June 1966), pp. 603–604. ISSN: 0037-7732. DOI: 10.1093/sf/44.4.603-a. eprint: <https://academic.oup.com/sf/article-pdf/44/4/603/6507927/44-4-603a.pdf>. URL: <https://doi.org/10.1093/sf/44.4.603-a>.
- [2] John P. A. Ioannidis. “Why Most Published Research Findings Are False”. In: *PLOS Medicine* 2.8 (Aug. 2005), null. DOI: 10.1371/journal.pmed.0020124. URL: <https://doi.org/10.1371/journal.pmed.0020124>.
- [3] Florian Prinz, Thomas Schlange, and Khusru Asadullah. “Believe it or not: how much can we rely on published data on potential drug targets?” In: *Nature Reviews Drug Discovery* 10.9 (Sept. 2011), pp. 712–712. ISSN: 1474-1784. DOI: 10.1038/nrd3439-c1. URL: <https://doi.org/10.1038/nrd3439-c1>.
- [4] C. Glenn Begley and Lee M. Ellis. “Raise standards for preclinical cancer research”. In: *Nature* 483.7391 (Mar. 2012), pp. 531–533. ISSN: 1476-4687. DOI: 10.1038/483531a. URL: <https://doi.org/10.1038/483531a>.
- [5] Monya Baker. “1, 500 scientists lift the lid on reproducibility”. In: *Nature* 533.7604 (May 2016), pp. 452–454. DOI: 10.1038/533452a. URL: <https://doi.org/10.1038/533452a>.
- [6] Gema Rodríguez-Pérez, Gregorio Robles, and Jesús M. González-Barahona. “Reproducibility and credibility in empirical software engineering: A case study based on a systematic literature review of the use of the SZZ algorithm”. In: *Information and Software Technology* 99 (July 2018), pp. 164–176. DOI: 10.1016/j.infsof.2018.03.009. URL: <https://doi.org/10.1016/j.infsof.2018.03.009>.
- [7] H. V. D. Sompel and C. Lagoze. “All aboard: toward a machine-friendly scholarly communication system”. In: *The Fourth Paradigm*. 2009.
- [8] 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>.
- [9] Mohamad Yaser Jaradeh et al. “Open Research Knowledge Graph: Towards Machine Actionability in Scholarly Communication”. In: (Jan. 2019).

- [10] Jeroen Bosman et al. “The Scholarly Commons - principles and practices to guide research communication”. In: (Sept. 2017). DOI: 10.31219/osf.io/6c2xt. URL: <https://doi.org/10.31219/osf.io/6c2xt>.
- [11] Hugo Alrøe and Egon Noe. “Second-Order Science of Interdisciplinary Research A Polyocular Framework for Wicked Problems”. In: *Constructivist Foundations* 10 (Nov. 2014), pp. 65–95.
- [12] Lutz Bornmann and Rüdiger Mutz. “Growth rates of modern science: A bibliometric analysis based on the number of publications and cited references”. In: *Journal of the Association for Information Science and Technology* 66.11 (Apr. 2015), pp. 2215–2222. DOI: 10.1002/asi.23329. URL: <https://doi.org/10.1002/asi.23329>.
- [13] Arif E. Jinha. “Article 50 million: an estimate of the number of scholarly articles in existence”. In: *Learned Publishing* 23.3 (July 2010), pp. 258–263. DOI: 10.1087/20100308. URL: <https://doi.org/10.1087/20100308>.
- [14] *Stencila*. <https://stenci.la/>. Accessed: 2021-05-09.
- [15] 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>.
- [16] John Willinsky. “Open Journal Systems”. In: *Library Hi Tech* 23.4 (Jan. 2005). Ed. by Scott P. Muir and Mark Leggott, pp. 504–519. ISSN: 0737-8831. DOI: 10.1108/07378830510636300. URL: <https://doi.org/10.1108/07378830510636300>.
- [17] Jon F. Claerbout and Martin Karrenbach. “Electronic documents give reproducible research a new meaning”. In: *SEG Technical Program Expanded Abstracts 1992*. 1992, pp. 601–604. DOI: 10.1190/1.1822162. URL: <https://app.dimensions.ai/details/publication/pub.1098913318>.
- [18] Bernard Lo. “Sharing Clinical Trial Data: Maximizing Benefits, Minimizing Risk”. In: *JAMA* 313.8 (Feb. 2015), pp. 793–794. ISSN: 0098-7484. DOI: 10.1001/jama.2015.292. eprint: <https://jamanetwork.com/journals/jama/articlepdf/2091787/jvp150006.pdf>. URL: <https://doi.org/10.1001/jama.2015.292>.
- [19] Enis Afgan et al. “The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update”. In: *Nucleic Acids Research* 44.W1 (May 2016), W3–W10. ISSN: 0305-1048. DOI: 10.1093/nar/gkw343. eprint: <https://academic.oup.com/nar/article-pdf/44/W1/W3/7633114/gkw343.pdf>. URL: <https://doi.org/10.1093/nar/gkw343>.
- [20] John Ioannidis and Chris Doucouliagos. “WHAT’S TO KNOW ABOUT THE CREDIBILITY OF EMPIRICAL ECONOMICS?” eng. In: *Journal of economic surveys* 27.5 (2013), pp. 997–1004. ISSN: 0950-0804.
- [21] Roger D. Peng, Francesca Dominici, and Scott L. Zeger. “Reproducible Epidemiologic Research”. In: *American Journal of Epidemiology* 163.9 (Mar. 2006), pp. 783–789. ISSN: 0002-9262. DOI: 10.1093/aje/kwj093. eprint: <https://academic.oup.com/aje/article-pdf/163/9/783/254266/kwj093.pdf>. URL: <https://doi.org/10.1093/aje/kwj093>.

- 
- [22] *National Science Foundation*. <https://www.nsf.gov/about/visit/>.
- [23] Steven N. Goodman, Daniele Fanelli, and John P. A. Ioannidis. “What does research reproducibility mean?” In: *Science Translational Medicine* 8.341 (2016), 341ps12–341ps12. DOI: 10.1126/scitranslmed.aaf5027.
- [24] . “Estimating the reproducibility of psychological science”. In: *Science* 349.6251 (2015). ISSN: 0036-8075. DOI: 10.1126/science.aac4716. eprint: <https://science.sciencemag.org/content/349/6251/aac4716.full.pdf>. URL: <https://science.sciencemag.org/content/349/6251/aac4716>.
- [25] 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>.
- [26] Pardo A. Hendler J. “A Primer on Machine Readability for Online Documents and Data”. In: 2019. URL: <https://www.data.gov/developers/blog/primer-machine-readability-online-documents-and-data>.
- [27] Joel T. Dudley and Atul J. Butte. “In silico research in the era of cloud computing”. In: *Nature Biotechnology* 28.11 (Nov. 2010), pp. 1181–1185. ISSN: 1546-1696. DOI: 10.1038/nbt1110-1181. URL: <https://doi.org/10.1038/nbt1110-1181>.
- [28] B. Howe. “Virtual Appliances, Cloud Computing, and Reproducible Research”. In: *Computing in Science Engineering* 14.4 (2012), pp. 36–41. DOI: 10.1109/MCSE.2012.62.
- [29] Ben Marwick. “Computational Reproducibility in Archaeological Research: Basic Principles and a Case Study of Their Implementation”. In: *Journal of Archaeological Method and Theory* 24.2 (June 2017), pp. 424–450. DOI: 10.1007/s10816-015-9272-9. URL: <https://doi.org/10.1007/s10816-015-9272-9>.
- [30] DeBruine Lisa M. Lakens D. “Improving Transparency, Falsifiability, and Rigour by Making Hypothesis Tests Machine Readable”. In: 2020. DOI: 10.31234/osf.io/5xcda. URL: <https://psyarxiv.com/5xcda/>.
- [31] Sören Auer and Sanjeet Mann. “Towards an Open Research Knowledge Graph”. In: *The Serials Librarian* 76.1-4 (2019), pp. 35–41. DOI: 10.1080/0361526X.2019.1540272. eprint: <https://doi.org/10.1080/0361526X.2019.1540272>. URL: <https://doi.org/10.1080/0361526X.2019.1540272>.
- [32] *Academic discipline*. [https://en.wikipedia.org/wiki/Academic\\_discipline](https://en.wikipedia.org/wiki/Academic_discipline).
- [33] *Google Scholar*. <https://scholar.google.com/>.
- [34] *The Web of Knowledge*. <http://login.webofknowledge.com/>.
- [35] Peter F. Lington et al. *Building Enterprise Systems with ODP*. Chapman and Hall/CRC, Sept. 2011. DOI: 10.1201/b11151. URL: <https://doi.org/10.1201/b11151>.
- [36] Jan Küenzl et al. “International Organization for Standardization (ISO)”. In: *International Encyclopedia of Civil Society*. Springer US, 2010, pp. 890–891. DOI: 10.1007/978-0-387-93996-4\_826. URL: [https://doi.org/10.1007/978-0-387-93996-4\\_826](https://doi.org/10.1007/978-0-387-93996-4_826).
- [37] “International Telecommunication Union”. In: *International Organization* 20.3 (1966), pp. 650–651. DOI: 10.1017/S0020818300012911.

- [38] M.W.A. Steen and J. Derrick. “ODP enterprise viewpoint specification”. In: *Computer Standards & Interfaces* 22.3 (Aug. 2000), pp. 165–189. DOI: 10.1016/s0920-5489(00)00031-3. URL: [https://doi.org/10.1016/s0920-5489\(00\)00031-3](https://doi.org/10.1016/s0920-5489(00)00031-3).
- [39] Francisco Durán, Manuel Roldán, and Antonio Vallecillo. “Using Maude to write and execute ODP information viewpoint specifications”. In: *Computer Standards & Interfaces* 27.6 (June 2005), pp. 597–620. DOI: 10.1016/j.csi.2004.10.008. URL: <https://doi.org/10.1016/j.csi.2004.10.008>.
- [40] R. Romero and A. Vallecillo. “Formalizing ODP computational viewpoint specifications in Maude”. In: *Proceedings. Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004. EDOC 2004*. IEEE. DOI: 10.1109/edoc.2004.1342517. URL: <https://doi.org/10.1109/edoc.2004.1342517>.
- [41] “Application Programming Interface (API)”. In: *Encyclopedia of Biometrics*. Springer US, 2009, pp. 41–41. DOI: 10.1007/978-0-387-73003-5\_858. URL: [https://doi.org/10.1007/978-0-387-73003-5\\_858](https://doi.org/10.1007/978-0-387-73003-5_858).
- [42] Hai-Quan Nguyen and Quan. NguyenHai. “Modeling the Engineering Viewpoint of ODP systems with MODERN”. In: 2005.
- [43] Kerry Raymond. “Reference Model of Open Distributed Processing (RM-ODP): Introduction”. In: *Open Distributed Processing*. Springer US, 1995, pp. 3–14. DOI: 10.1007/978-0-387-34882-7\_1. URL: [https://doi.org/10.1007/978-0-387-34882-7\\_1](https://doi.org/10.1007/978-0-387-34882-7_1).
- [44] Xiaoli Chen et al. “CERN Analysis Preservation: A Novel Digital Library Service to Enable Reusable and Reproducible Research”. In: *Research and Advanced Technology for Digital Libraries*. Ed. by Norbert Fuhr et al. Cham: Springer International Publishing, 2016, pp. 347–356. ISBN: 978-3-319-43997-6.
- [45] Donald H. Perkins. *Introduction to High Energy Physics*. 4th ed. Cambridge University Press, 2000. DOI: 10.1017/CB09780511809040.
- [46] J. Kuncar, L. Nielsen, and T. Simko. “Invenio v2.0: A Pythonic Framework for Large-Scale Digital Libraries”. In: 2014.
- [47] Brian F. Lavoie. “The Open Archival Information System Reference Model: Introductory Guide”. In: 33.2 (2004), pp. 68–81. DOI: doi : 10.1515/MFIR.2004.68. URL: <https://doi.org/10.1515/MFIR.2004.68>.
- [48] AJ Peters, EA Sindrilaru, and G Adde. “EOS as the present and future solution for data storage at CERN”. In: *Journal of Physics: Conference Series* 664.4 (Dec. 2015), p. 042042. DOI: 10.1088/1742-6596/664/4/042042. URL: <https://doi.org/10.1088/1742-6596/664/4/042042>.
- [49] E. Cano et al. “The new CERN tape software - getting ready for total performance”. In: *J. Phys. Conf. Ser.* 664.4 (2015), p. 042007. DOI: 10.1088/1742-6596/664/4/042007.
- [50] Peter L. Jones and Nils Høimyr. “TWiki a Collaboration Tool for the LHC”. In: WikiSym ’11. Mountain View, California: Association for Computing Machinery, 2011, pp. 207–208. ISBN: 9781450309097. DOI: 10.1145/2038558.2038596. URL: <https://doi.org/10.1145/2038558.2038596>.

- 
- [51] The CMS Collaboration, S Chatrchyan, and G Hmayakyan. “The CMS experiment at the CERN LHC”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08004–S08004. DOI: 10.1088/1748-0221/3/08/S08004. URL: <https://doi.org/10.1088/1748-0221/3/08/S08004>.
- [52] Martin Vesely et al. “CERN document server: Document management system for grey literature in a networked environment”. In: *Publishing Research Quarterly* 20.1 (Mar. 2004), pp. 77–83. ISSN: 1936-4792. DOI: 10.1007/BF02910863. URL: <https://doi.org/10.1007/BF02910863>.
- [53] Bertram Ludäscher et al. “Capturing the ”Whole Tale” of Computational Research: Reproducibility in Computing Environments”. In: (Oct. 2016).
- [54] Adam Brinckman et al. “Computing environments for reproducibility: Capturing the “Whole Tale””. In: *Future Generation Computer Systems* 94 (2019), pp. 854–867. ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2017.12.029>. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X17310695>.
- [55] René Peinl, Florian Holzschuher, and Florian Pfitzer. “Docker Cluster Management for the Cloud - Survey Results and Own Solution”. In: *Journal of Grid Computing* 14.2 (June 2016), pp. 265–282. ISSN: 1572-9184. DOI: 10.1007/s10723-016-9366-y. URL: <https://doi.org/10.1007/s10723-016-9366-y>.
- [56] Yang Cao et al. “DataONE: A Data Federation with Provenance Support”. In: IPAW 2016. McLean, VA, USA: Springer-Verlag, 2016, pp. 230–234. ISBN: 9783319405926.
- [57] Rachana Ananthakrishnan et al. “Globus Platform Services for Data Publication”. In: PEARC ’18. Pittsburgh, PA, USA: Association for Computing Machinery, 2018. ISBN: 9781450364461. DOI: 10.1145/3219104.3219127. URL: <https://doi.org/10.1145/3219104.3219127>.
- [58] *Girder, 2017*. <https://girder.readthedocs.io/en/latest/>. Accessed: 2017-03-01.
- [59] *First Steps with Celery, 2018-2019*. <https://docs.celeryproject.org/en/stable/getting-started/first-steps-with-celery.html>. Accessed: 2021-05-07.
- [60] *Using Redis, 2018-2019*. <https://docs.celeryproject.org/en/stable/getting-started/brokers/redis.html>. Accessed: 2021-05-07.
- [61] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA, 2015. URL: <http://www.rstudio.com/>.
- [62] *Jupyter Notebook, 2015*. <https://jupyter-notebook.readthedocs.io/en/stable/>. Accessed: 2021-05-09.
- [63] *EmberJS*. <https://emberjs.com/>. Accessed: 2017-03-01.
- [64] *eLife Magazine*. <https://elifesciences.org/>. Accessed: 2017-03-01.
- [65] *Binder*. <https://mybinder.org/>. Accessed: 2021-05-08.
- [66] Michael Aufreiter, Alkesandra Pawlik, and Nokome Bentley. “Stencila – an office suite for reproducible research”. In: *eLife Magazine* 2.3 (July 2018), p. 120450. ISSN: 2050-084X. URL: <https://elifesciences.org/labs/c496b8bb/stencila-an-office-suite-for-reproducible-research>.

- [67] *Journal Article Tag Suite*. <http://jats.nlm.nih.gov/1.2/>. Accessed: 2021-05-03.
- [68] “Replication study: Transcriptional amplification in tumor cells with elevated c-Myc”. English (US). In: *eLife* 7 (Jan. 2018). Publisher Copyright: © Lewis et al. Copyright: Copyright 2018 Elsevier B.V., All rights reserved. ISSN: 2050-084X. DOI: 10.7554/eLife.30274.
- [69] *D4Science Infrastructure*. <https://www.d4science.org/>. Accessed: 2021-05-09.
- [70] *Framework Programmes for Research and Technological Development*. [https://en.wikipedia.org/wiki/Framework\\_Programmes\\_for\\_Research\\_and\\_Technological\\_Development](https://en.wikipedia.org/wiki/Framework_Programmes_for_Research_and_Technological_Development). Accessed: 2021-05-01.
- [71] *Geant Project*. <https://geant3plus.archive.geant.net/Pages/home.html>. Accessed: 2021-05-06.
- [72] *Enabling Grids for E-Science (EGEE)*. <https://eu-egee-org.web.cern.ch/index.html>. Accessed: 2021-05-06.
- [73] *Diligent*. <https://diligent.com/>. Accessed: 2021-05-06.
- [74] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. “Virtual Research Environments: An Overview and a Research Agenda”. In: *Data Science Journal* 12.0 (2013), GRDI75–GRDI81. DOI: 10.2481/dsj.grdi-013. URL: <https://doi.org/10.2481/dsj.grdi-013>.
- [75] Leonardo Candela, Donatella Castelli, and Pasquale Pagano. *gCube v1.0: A Software System for Hybrid Data Infrastructures*. Jan. 2008.
- [76] *pan-European Grid middleware (gLite)*. <https://glite.web.cern.ch/glite/>. Accessed: 2021-05-01.
- [77] *Liferay portal technology*. <https://www.liferay.com/>. Accessed: 2021-05-02.
- [78] Tyler Phillips et al. “AuthN-AuthZ: Integrated, User-Friendly and Privacy-Preserving Authentication and Authorization”. In: *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. 2020, pp. 189–198. DOI: 10.1109/TPS-ISA50397.2020.00034.
- [79] Massimiliano Assante et al. “Enacting open science by D4Science”. In: *Future Generation Computer Systems* 101 (May 2019). DOI: 10.1016/j.future.2019.05.063.
- [80] Massimiliano Assante et al. “The gCube system: Delivering Virtual Research Environments as-a-Service”. In: *Future Generation Computer Systems* 95 (June 2019), pp. 445–453. DOI: 10.1016/j.future.2018.10.035. URL: <https://doi.org/10.1016/j.future.2018.10.035>.
- [81] Stencila Community. *Stencila*. Version v0.34.3. Dec. 2020. URL: <https://openbase.com/js/stencila/documentation>.
- [82] *Stencila Hub*. <https://hub.stencila/>. Accessed: 2021-05-09.
- [83] *JavaScript*. <https://www.javascript.com/>. Accessed: 2021-05-09.
- [84] *Python*. <https://www.python.org/>. Accessed: 2021-05-09.
- [85] *R Project*. <https://www.r-project.org/>. Accessed: 2021-05-09.
- [86] *Rust*. <https://www.rust-lang.org/>. Accessed: 2021-05-09.

- 
- [87] *TypeScript*. <https://www.typescriptlang.org/docs/handbook/project-references.html>. Accessed: 2021-05-09.
- [88] *Node.js*. <https://nodejs.org/en/docs/>. Accessed: 2021-05-09.
- [89] ORKG Team. “Welcome to ORKG’s python package documentation!” In: Read the Docs, 2020. URL: <https://orkg.readthedocs.io/en/latest/index.html>.
- [90] ORKG Team. “ORKG Documentation”. In: Technische Informationsbibliothek (TIB), 2020. URL: <https://projects.tib.eu/orkg/documentation/>.
- [91] *ORKG Python Package*. <https://orkg.readthedocs.io/en/latest/introduction.html>. Accessed: 2021-05-04.
- [92] Brian Edgar and John Willinsky. “A Survey of the Scholarly Journals Using Open Journal Systems”. In: *OJS på dansk* 1 (Sept. 2010). DOI: 10.7146/ojssb.v1i1.2707.
- [93] PKP Community. “Open Journal Systems”. In: Public Knowledge Project (PKP), 2021. URL: <https://pkp.sfu.ca/ojs/>.
- [94] OJS Team. “OJS Documentation”. In: Public Knowledge Project (PKP), 2020. URL: [https://docs.pkp.sfu.ca/ojs-2-technical-reference/en/about\\_open\\_journal\\_systems](https://docs.pkp.sfu.ca/ojs-2-technical-reference/en/about_open_journal_systems).
- [95] OJS Team. “Open journal systems user guide”. In: Open journal systems (OJS), 2021. URL: <https://openjournalssystem.com/ojs-3-user-guide/>.
- [96] Executable Books Community. *Jupyter Book*. Version v0.10. Feb. 2020. DOI: 10.5281/zenodo.4539666. URL: <https://doi.org/10.5281/zenodo.4539666>.
- [97] Saba Haddad et al. “Iron-regulatory proteins secure iron availability in cardiomyocytes to prevent heart failure”. In: *European Heart Journal* 38.5 (Aug. 2016), pp. 362–372. ISSN: 0195-668X. DOI: 10.1093/eurheartj/ehw333. eprint: <https://academic.oup.com/eurheartj/article-pdf/38/5/362/10372887/ehw333.pdf>. URL: <https://doi.org/10.1093/eurheartj/ehw333>.
- [98] *ThebeLab*. <https://thebe.readthedocs.io/en/latest/>. Accessed: 2021-05-07.
- [99] *Google Colab*. <https://colab.research.google.com/notebooks/intro.ipynb>. Accessed: 2021-05-07.
- [100] *Anaconda Software Distribution*. Version Vers. 2-2.4.0. 2020. URL: <https://docs.anaconda.com/>.
- [101] Norman Gentsch et al. “Catch crop diversity increases rhizosphere carbon input and soil microbial biomass”. In: *Biology and Fertility of Soils* 56.7 (Oct. 2020), pp. 943–957. ISSN: 1432-0789. DOI: 10.1007/s00374-020-01475-8. URL: <https://doi.org/10.1007/s00374-020-01475-8>.
- [102] JetBrains Community. *PyCharm: The Python IDE for Professional Developers*. JetBrains, s.r.o. Czech Republic, Kavčí Hory Office Park, Na Hřebenech II 1718/10 Praha 4 - Nusle - 140 00, 2021. URL: <https://www.jetbrains.com/pycharm/>.
- [103] *Spyder IDE*. <https://www.spyder-ide.org/>. Accessed: 2021-05-01.
- [104] *Atom IDE*. <https://atom.io/>. Accessed: 2021-05-01.



- [105] Francesco Russo, Dario Righelli, and Claudia Angelini. “Advantages and Limits in the Adoption of Reproducible Research and R-Tools for the Analysis of Omic Data”. In: *Computational Intelligence Methods for Bioinformatics and Biostatistics*. Ed. by Claudia Angelini, Paola MV Rancoita, and Stefano Rovetta. Cham: Springer International Publishing, 2016, pp. 245–258. ISBN: 978-3-319-44332-4.
- [106] Roger D. Peng. “Reproducible research in computational science”. eng. In: *Science (New York, N.Y.)* 334.6060 (Dec. 2011). 334/6060/1226[PII], pp. 1226–1227. ISSN: 1095-9203. DOI: 10.1126/science.1213847. URL: <https://doi.org/10.1126/science.1213847>.
- [107] *Gallery of Jupyter Books*. <https://executablebooks.org/en/latest/gallery.html>. Accessed: 2021-05-01.
- [108] *MalariaGen Book*. <https://malariagen.github.io/vector-data/landing-page.html>. Accessed: 2021-05-01.
- [109] *TensorDiffEq Book*. <https://docs.tensordiffeq.io/>. Accessed: 2021-05-01.
- [110] Cindy Judd et al. “Google Docs: A Review”. In: *Against the Grain* 20 (Jan. 2009), pp. 14–17. DOI: 10.7771/2380-176X.2736.
- [111] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: (2020). arXiv: 2005.14165 [cs.CL].
- [112] Luciano Floridi and Massimo Chiriatti. “GPT-3: Its Nature, Scope, Limits, and Consequences”. In: *Minds and Machines* 30 (), pp. 1–14. DOI: 10.1007/s11023-020-09548-1.
- [113] Ke-Li Chiu and Rohan Alexander. *Detecting Hate Speech with GPT-3*. Mar. 2021.
- [114] Ioannis Mollas et al. *ETHOS: an Online Hate Speech Detection Dataset*. arXiv: 2006.08328.
- [115] Antonios Anagnostou, Ioannis Mollas, and Grigorios Tsoumakas. “Hatebusters: A Web Application for Actively Reporting YouTube Hate Speech”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, July 2018, pp. 5796–5798. DOI: 10.24963/ijcai.2018/841. URL: <https://doi.org/10.24963/ijcai.2018/841>.
- [116] Jason Baumgartner et al. “The Pushshift Reddit Dataset”. In: *Proceedings of the International AAAI Conference on Web and Social Media* 14.1 (May 2020), pp. 830–839. URL: <https://ojs.aaai.org/index.php/ICWSM/article/view/7347>.
- [117] Jesse M. Alston and Jessica A. Rick. “A Beginner’s Guide to Conducting Reproducible Research”. In: *The Bulletin of the Ecological Society of America* (Jan. 2021). DOI: 10.1002/bes2.1801. URL: <https://doi.org/10.1002/bes2.1801>.
- [118] John P. A. Ioannidis. “How to Make More Published Research True”. In: *PLoS Medicine* 11.10 (Oct. 2014), e1001747. DOI: 10.1371/journal.pmed.1001747. URL: <https://doi.org/10.1371/journal.pmed.1001747>.
- [119] John P. A. Ioannidis et al. “Meta-research: Evaluation and Improvement of Research Methods and Practices”. In: *PLOS Biology* 13.10 (Oct. 2015), e1002264. DOI: 10.1371/journal.pbio.1002264. URL: <https://doi.org/10.1371/journal.pbio.1002264>.
- [120] Paul E. Smaldino and Richard McElreath. “The natural selection of bad science”. In: *Royal Society Open Science* 3.9 (Sept. 2016), p. 160384. DOI: 10.1098/rsos.160384. URL: <https://doi.org/10.1098/rsos.160384>.