# Packaging research data with RO-Crate

*This manuscript ([permalink](#)) was automatically generated from [stain/ro-crate-paper@2c3f404](#) on May 14, 2021.*

## Authors

- **Stian Soiland-Reyes**
  [0000-0001-9842-9718](#) · [stain](#) · [soilandreyes](#)
  Department of Computer Science, The University of Manchester, UK; Informatics Institute, Faculty of Science, University of Amsterdam, NL · Funded by [BioExcel-2](#) (European Commission [H2020-INFRAEDI-2018-1 823830](#)); [IBISBA](#) (H2020-INFRAIA-2017-1-two-stage 730976)

- **Peter Sefton**
  [0000-0002-3545-944X](#) · [ptsefton](#)
  Faculty of Science, University Technology Sydney, Australia

- **Simone Leo**
  [0000-0001-8271-5429](#) · [simleo](#) · *[simleo](#)*
  Center for Advanced Studies, Research, and Development in Sardinia (CRS4), Pula (CA) , Italy · Funded by [EOSC-Life](#) (European Commission [H2020-INFRAEOSC-2018-2 824087](#))

- **RO-Crate Community**
  [https://www.researchobject.org/ro-crate/community](https://www.researchobject.org/ro-crate/community) · [researchobject](#)

- **Carole Goble**
  [0000-0003-1219-2137](#) · [carolegoble](#)
  Department of Computer Science, The University of Manchester, UK · Funded by [BioExcel-2](#) (European Commission [H2020-INFRAEDI-2018-1 823830](#)); [EOSC-Life](#) (European Commission [H2020-INFRAEOSC-2018-2 824087](#)); [IBISBA](#) (European Commission [H2020-INFRAIA-2017-1-two-stage 730976](#), [H2020-INFRADEV-2019-2 871118](#)); [SyntheSys+](#) (European Commission [H2020-INFRAIA-2018-1 823827](#))

- **Paul Groth**
  [0000-0003-0183-6910](#) · [pgroth](#)
  Informatics Institute, Faculty of Science, University of Amsterdam, NL

RO-Crate is an open, community-driven, and lightweight approach to packaging research data with their metadata . Based on schema.org annotations in JSON-LD, RO-Crate aims to establish best-practice for formal metadata description accessible and practical for use in a wide variety of situations.

An RO-Crate is a structured archive of all the items that contributed to a research outcome, including their identifiers, provenance, relations and annotations. As a general purpose packaging framework for data and their metadata, RO-Crate is used across multiple areas, including bioinformatics, digital humanities and regulatory submissions. By applying "just enough" Linked Data standards, RO-Crate simplifies the process of making research outputs FAIR.

# Introduction

The move towards open science and open research practices has increased the demand for the publication of more artefacts of the research process [**???**]. This is particularly apparent in domains

that rely on computational experiments; for example, the publication of software, datasets and records of the dependencies that such experiments rely on [1].

It is often argued that the publication of these assets, and specifically software [2] and data, should follow the The FAIR principles [3]; namely, that they are Findable, Accessible, Interoperable and Reusable. These principles are agnostic to the *implementation* strategy needed to comply with them. Hence, there has been an increasing amount of work in the development of systems and specifications that aim to fulfil these goals [4]. Important examples include data publication with rich metadata (e.g. Zenodo [5]), domain specific data deposition (e.g. PDB [6]) and following practices for reproducible research software [7] (e.g. use of containers).

These strategies are focused primarily on one *type* of artefact. To address this, [8] introduced the notion of **research objects** – *semantically rich aggregations of (potentially distributed) resources that provide a layer of structure on top of information delivered as Linked Data*. A research object combines the ability to bundle multiple types of artefacts together; for example, CSV files, code, examples, and figures. This provides a compelling vision as an approach for implementing FAIR. However, existing research object implementations require a large technology stack, are tailored to a particular platform and are also not easily usable by end-users.

To address this gap, a new community came together [9] to develop **RO-Crate** - a *lightweight approach to packaging and aggregating research artefacts with their metadata*. The aim of this paper is to introduce RO-Crate and assess it as a strategy for making multiple types of research artefacts FAIR. Specifically, the contributions of this paper are as follows:

1. an introduction to RO-Crate, its purpose and context;
2. a guide to the RO-Crate community and tooling;
3. and exemplar usage of RO-Crate for different artefacts and its use as a connective tissue for such artefacts.

The rest of this paper is organised as follows. We first describe RO-Crate, the assumptions underlying it, and define RO-Crate technically and formally. We then proceed to introduce the community and tooling. We then analyse RO-Crate with respect to usage in a diverse set of domains. Finally, we present related work and conclude.

# RO-Crate

As previously stated, RO-Crate provides a lightweight approach to packaging research artifacts with their metadata. What does that mean? Imagine a research paper reporting on the sequence analysis of proteins in an experiment on mice. The sequence output files, the sequence analysis code, and reports summarising statistical measures or outputs are all important and inter-related research outputs and consequently would all be put in a directory.

The question then arises as to how the directory with all this material should be packaged in a manner that is accessible and usable by others. By usable we mean not just readable by humans but accessible programmatically. Fundamentally, how can one easily get these data and work with them. A de facto approach to sharing such compilations is through the use of a compressed archive (e.g. a zip file). While this solves the problem of "packaging", it does not imply that the downstream user can *easily* access the data in a programmatic fashion, or know what role each file plays in the research. This leads to the need for explicit metadata about the contents of the folder.

Examples of metadata descriptions across a wide range of domains abound within the literature, both in research data management (?cite) and within library and information systems (?cite). However,

many of these approaches (discussed in the related work section) require knowledge of metadata schemas, particular annotation systems, or the use of obscure or complex software stacks. Indeed, particularly within research, these requirements have led to a lack of adoption and growing frustration with current tooling and specifications [**???**].

RO-Crate seeks to address this complexity by:

1. being easy to understand and simple conceptually;
2. providing a strong and opinionated guide to current best practices;
3. adopting software stacks that are widely used on the Web.

In the following sections we show how the RO-Crate specification and ecosystem achieve these goals, which concur in forming our definition of "lightweight".

# Conceptual Definition

A key premise of RO-Crate is the existence of a wide variety of resources on the Web that can help describe research. As such, RO-Crate relies on concepts from the Web, in particular that of Linked Data.
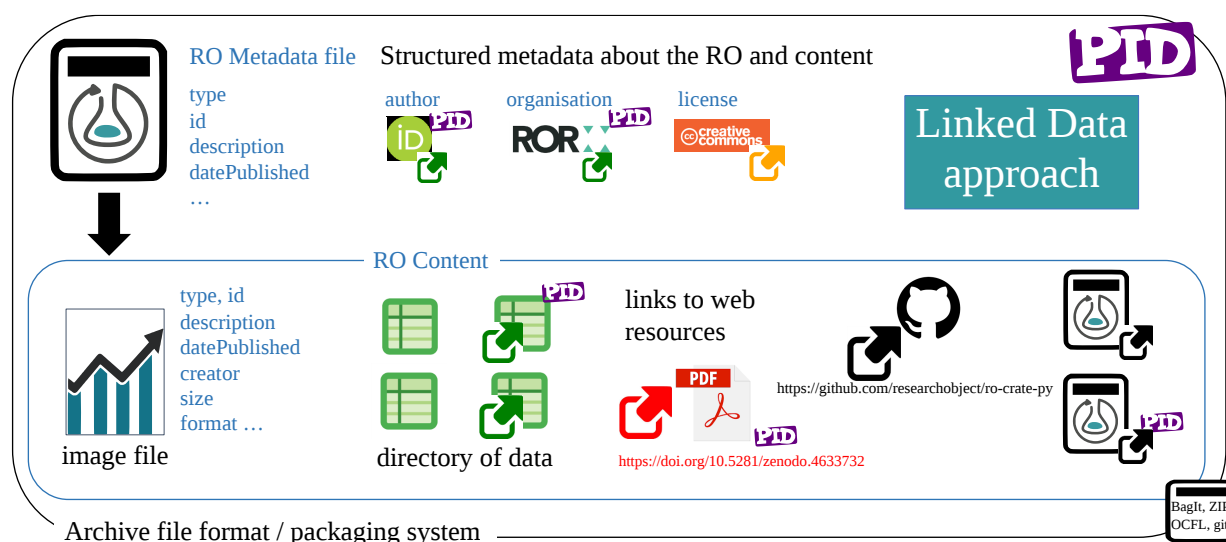


**Figure 1: Conceptual overview of RO-Crate**. A *Persistent Identifier* (PID) identifies a *Research Object* (RO), which is archived using BagIt[10], OCFL[**???**], git or ZIP. Using Linked Data, the RO is described within a , providing identifiers for authors using ORCID, organizations using ROR and licenses such as Creative Commons. The *RO-Crate content* is further described with its own metadata. Data can be embedded files and directories, as well as links to external web resources, PIDs and nested RO-Crates.

## Linked Data as a core principle

Linked Data [11] is a core principle of RO-Crate, and IRIs[1] are therefore used to identify the RO-Crate, its constituent parts and metadata descriptions, and the properties and classes used in the metadata.

Using Linked Data, where consumers can follow the links for more (ideally both human- or machine-readable) information, the RO-Crate can then be sufficiently *self-describing* and entities interrelated using global identifiers, without needing to recursively fully describe every referenced entity.

The foundation on Linked Data and shared vocabularies also means multiple RO-Crates and other Linked Data resources can be indexed, combined, queried, integrated or transformed using existing Semantic Web technologies like SPARQL and knowledge graph triple stores.

# RO-Crate is a self-described container

An RO-Crate is defined as a self-described **Root Data Entity** that describes and contains *data entities*, which are further described using *contextual entities*. A **data entity** is primarily a *file* (bytes on disk somewhere) or a *directory* (dataset of named files) — while a **contextual entity** exists outside the information system (e.g. a Person) and its bytes are primarily metadata.

The Root Data Entity is a directory, the RO-Crate root, identified by the presence of the **RO-Crate Metadata File** with the fixed name `ro-crate-metadata.json`. This is a JSON-LD file that describes the RO-Crate, its content and related metadata using Linked Data.

The minimal [requirements for the root data entity metadata](#) are `name`, `description` and `datePublished`, as well as a contextual entity identifying its `license` — additional metadata is frequently added depending on the purpose of the particular RO-Crate.

Because the RO-Crate is just a set of related resources and datasets, it can be stored, transferred or published in multiple ways, such as BagIt [10], Oxford Common File Layout [**???**] (OCFL), downloadable ZIP archives in Zenodo or through dedicated online repositories, as well as published directly on the Web, e.g. using GitHub Pages. Combined with Linked Data identifiers, this caters for a diverse set of storage and access requirements across different scientific domains, from metagenomics workflows producing hundreds of gigabytes of genome data to cultural heritage records with access restrictions for personally identifiable data.

# Data Entities are described using Contextual Entities

RO-Crate distinguishes between [data and contextual entities](#) in a similar way to HTTP terminology's early attempt to separate *information* (data) and *non-information* (contextual) resource [**???**]. While data entities, in the most common scenario, are files and directories located by relative IRI references within the RO-Crate Root, they can also be Web resources identified with absolute http/https IRIs.

As both types of entities are identified by IRIs, their distinction is allowed to be blurry; data entities can be located anywhere and be complex, and contextual entities can have a Web presence beyond their description inside the RO-Crate. For instance `https://orcid.org/0000-0002-1825-0097` is primarily an identifier for a person, but secondarily also a Web page that describes that person and their academic work.

Any particular IRI might appear as a contextual entity in one RO-Crate and a data entity in another; their distinction lies in the fact that data entities can be considered to be *contained* or captured by the RO-Crate, while contextual entities mainly *explain* the RO-Crate and its entities. It follows that an RO-Crate should have one or more data entities, although this is not a formal requirement.

Figure [2](#) shows a UML view of RO-Crate , highlighting the different types of data entities and contextual entities that can be aggregated and related.

**Figure 2: UML model view of RO-Crate.** The *RO-Crate Metadata File* conforms to a version of the specification, which mainly describes the *RO-Crate Root Data Entity* representing the Research Object as a dataset. The RO-Crate aggregates *data entities* (`hasPart`) which are further described using *contextual entities*. Multiple types and relations from schema.org allow annotations to be more specific, including figures, nested datasets, computational workflows, people, organizations, instruments and places.

# Best Practice Guide rather than strict specification

RO-Crate builds on Linked Data standards and community-evolved vocabularies like JSON-LD [**???**] and [schema.org](), but rather than enforce additional constraints and limits — using perhaps intimidating technologies like *RDF shapes* (SHACL, ShEx) — RO-Crate aims instead to build a set of best practice guides on how to apply existing standards in a common way to describe research outputs and their provenance.

As such, the [RO-Crate 1.1]() specification [13] can be seen as an opinionated and example-driven guide to writing schema.org metadata as JSON-LD, which leaves it open for implementers to include additional metadata using other schema.org types and properties, or even additional Linked Data vocabularies/ontologies or their own ad-hoc terms.

This does not mean, however, that RO-Crate does not have constraints. Crucially, the specification is quite strict on the JSON-LD being *flattened* and *compacted*, to facilitate lightweight consumption and generation of the RO-Crate Metadata file without the need for RDF libraries. In addition, work has now begun to formalised different *profiles* of RO-Crates, which may impose additional constraints based on the needs of a specific domain or use case. For instance, submission to a workflow repository would expect an RO-Crate to contain at least one workflow with a declared license and workflow language; specific additional recommendations could also be needed to meet the emerging requirements of [FAIR Digital Objects](). These optional profiles are planned to build on a combination of JSON Schemas and RDF shapes, but again these will primarily provide developers with guidance and suggestions rather than strict validation requirements.

# Checking Simplicity

As previously stated, one aim of RO-Crate is to be conceptually simple. This simplicity has been repeatedly checked and confirmed through a community review process. See for example the discussion at ([GitHub issue #71 on ad-hoc vocabularies]()).

To further verify this idea, we have here formalised the RO-Crate definition (see Appendix X). An important result of this exercise is that the underlying data structure of RO-Crate is a depth limited tree of 1 level. The formalisation also emphasises the *boundedness* of the structure; namely, the fact that elements are specifically identified as being either semantically *contained* by the RO-Crate (`hasPart`) or mainly referenced (`mentions`) and typed as *external* to the Research Object (Contextual Entities).

# Technical implementation of the model

The above conceptual model has been realised using JSON-LD and schema.org in a prescriptive form as discussed in the best practice approach. This technical approach again caters for simplicity.

JSON-LD provides a way to express Linked Data as a JSON structure, where a *context* provides mapping to RDF properties and classes. While JSON-LD can't map arbitrary JSON structure to RDF, we found it does provide a good starting point for making a JSON-based language that is also interpretable as Linked Data.

In the design of RO-Crate we did however find that JSON-LD alone, like many RDF technologies, has too many degrees of freedom and hidden complexities for software developers to reliably produce and consume without specialized expertise or software libraries. A large part of the RO-Crate specification is therefore dedicated to describing JSON structures.

## RO-Crate JSON-LD

RO-Crate mandates the use of flattened, compacted JSON-LD where a single `@graph` array contains all the data and contextual entities in a flat list:

```
{ "@context": "https://w3id.org/ro/crate/1.1/context",
  "@graph": [
    { "@id": "ro-crate-metadata.json",
      "@type": "CreativeWork",
      "about": {"@id": "./"},
    },
    { "@id": "./",
      "@type": "Dataset",
      "hasPart": [
        {"@id": "data1.txt"},
        {"@id": "data2.txt"}
      ]
    },
    { "@id": "data1.txt",
      "@type": "File",
      "author": {"@id": "#alice"},
    },
    { "@id": "data2.txt",
      "@type": "File",
      "author": {"@id": "#alice"},
    },
    { "@id": "#alice",
      "@type": "Person",
      "name": "Alice",
    },
    { "@id": "http://sws.geonames.org/8152662/",
      "@type": "Place",
      "name": "Catalina Park"
    }
] }
```

*Figure X: Simplified RO-Crate JSON-LD showing the flattened compacted `@graph` array*

It can be argued that this is a more graph-like approach than the tree structure JSON would otherwise invite, and which is normally emphasised as a feature of JSON-LD in order to "hide" its RDF nature.

However, we found that the use of trees for, say, *a Person entity appearing as author of a* `File` *which nests under a* `Dataset`, `hasPart`, counter-intuitively leads one to consider the JSON-LD as an RDF Graph, as an identified Person entity can then appear at multiple and repeated points of the tree (e.g. author of multiple files), necessitating node merging or duplication.

By comparison, a single flat `@graph` array approach means that applications can process and edit each entity as pure JSON by a simple lookup based on `@id`. At the same time, lifting all entities to the same level emphasizes Research Object's principle that describing the context and provenance is just as important as describing the data.

While RO-Crate mainly uses schema.org, we found that JSON-based RO-Crate applications that are largely unaware of JSON-LD still may want to process the `@context` to find Linked Data definitions of unknown properties and types. Thus, RO-Crate provides its own versioned JSON-LD context, which has a similar flat list with the mapping from JSON-LD keys to their URI equivalents; for instance, `author` maps to [http://schema.org/author](http://schema.org/author). Not reusing the official schema.org context means RO-Crate is also able to map in additional vocabularies where needed, namely the *Portland Common Data Model* (PCDM) [**???**] for repositories and Bioschemas [**???**] for describing computational workflows.

Similarly, rather than relying on implications from `"@type: @id"` annotations in the context, RO-Crate JSON-LD distinguishes explicitly between references to other entities (`{"id": "#alice"}`) and string values (`"Alice"`) - meaning RO-Crate applications can find the corresponding entity without parsing the `@context`.

# RO-Crate Community

The RO-Crate conceptual model, implementation and best practice guides are developed by a growing community. The RO-Crate community is a key aspect of its effectiveness in making research artefacts FAIR. Fundamentally, the community provides the overall context of the implementation and model and ensures its interoperability.

The RO-Crate community consists of:

1. a diverse set of people representing a variety of stakeholders;
2. a set of collective norms;
3. an open platform that facilitates communication (GitHub, Google Docs, monthly telcons).

## People

The initial concept of RO-Crate was formed at the first Workshop on Research Objects ([RO2018](#)) held as part of the IEEE conference on eScience. This, in turn, followed up considerations made at an [RDA meeting on Research Data Packaging](#) that found similar goals across multiple data packaging efforts [9]: simplicity, structured metadata and the use of JSON-LD.

Discussions at RO2018 identified that the original Wf4Ever Research Object ontologies [14], while in principle sufficient for packaging research artefacts with rich descriptions, were, in practice, considered inaccessible for regular programmers (e.g. Web developers) and in danger of being incomprehensible for domain scientists due to their reliance on Semantic Web technologies and other ontologies.

DataCrate [15] was presented at RO2018 as a promising lightweight alternative approach, and an agreement was made by a group of volunteers to attempt building "RO Lite" as a combination of DataCrate's implementation and Research Object's principles.

This group, originally made up of library and semantic web experts, has subsequently grown to include domain scientists, developers, publishers and more. This perspective of multiple views led to the specification being used in a variety of domains, from bioinformatics and regulatory submissions to humanities and cultural heritage preservation.

The RO-Crate community is strongly engaged with the European-wide biology/bioinformatics collaborative e-Infrastructure, ELIXIR, [16], along with European Open Science Cloud (EOSC) projects

including EOSC-Life and FAIRplus. RO-Crate has also established collaborations with Bioschemas, GA4GH, OpenAIRE and multiple H2020 projects.

A key set of stakeholders is that of developers; the community has made a point of attracting developers that are able to implement the specifications but, importantly, keeps "developer user experience" in mind. This means that the specifications are straightforward to implement and thus do not require expertise in technologies that are not widely deployed.

This notion of catering to "developer user experience" is an example of the set of norms that have developed and now define the community.

## Norms

We use the notion of norm here instead of principle: these are conventions or notions that are prevalent within the community but not formalized. Here, we distil what we as authors believe are the critical set of norms that have facilitated the development of RO-Crate and contribute to the ability for RO-Crate research packages to be FAIR. This is not to say that there are no other norms within the community or that every one in the community holds these uniformly. Instead, what we emphasize is that these norms are helpful and also shaped by community practice.

1. Simplicity
2. Developer friendliness
3. Focus on examples and best practice over rigorous specification
4. Reuse "just enough" Web standards

A core norm of RO-Crate is that of **simplicity**, which sets the scene for how we guide developers to structure metadata with RO-Crate. We focus mainly on documenting simple approaches to the most common use cases, such as authors having an affiliation. This norm of simplicity also influences our take on **developer friendliness**; for instance, in that we are using the Web-native JSON format and allowing only a few of JSON-LD's flexible Linked Data features, and that the RO-Crate documentation is largely built up by **examples** showcasing **best practice**, rather than rigorous specifications. In a sense, we are allowed to do this by building on existing **Web standards** that themselves are defined rigorously, which we utilise ***"just enough"*** in order to benefit from the advantages of Linked Data (e.g. extensions by namespaced vocabularies), without imposing too many developer choices or uncertainties (e.g. having to choose between the many RDF syntaxes).

RO-Crate is not developed in a vacuum, and while the above norms alone could easily lead to the creation of "yet another" JSON format, we are also keeping the goal of **FAIR interoperability** of the captured metadata, and therefore follow closely FAIR best practices and current developments such as data citations, persistent identifiers, open repositories and recommendations for sharing research outputs and software.

## Open Platforms

The critical infrastructure that enables the community around RO-Crate is the use of open development platforms. This might not seem novel, but it underpins the importance of open community access to supporting FAIR. Specifically, it is difficult to build and consume FAIR research artefacts without being able to access the specifications, understand how they are developed, knowing about any potential implementation issues, and discuss usage to evolve best practices.

It was seen as more important to document real-life examples and best practice guides than to develop a rigorous specification. At the same time, we agreed to be opinionated on the syntactic form

to reduce the jungle of implementation choices; we wanted to keep the important aspects of Linked Data to adhere to the FAIR principles while retaining the option of combining and extending the structured metadata using the existing Semantic Web stack, not just build "yet another" standalone JSON format.

Further work during 2019 started adapting the DataCrate documentation through a more collaborative and exploratory *RO-Lite* phase, initially using Google Docs for review and discussion, then moving to GitHub as a collaboration space for developing what is now the RO-Crate specification, maintained as Markdown in GitHub Pages and published through Zenodo.

In addition to the typical Open Source-style development with GitHub issues and pull requests, the RO-Crate Community now has two regular monthly calls, a Slack channel and mailing list for coordinating the project, and many of its participants collaborate on RO-Crate at multiple conferences and coding events such as the ELIXIR BioHackathon. The community is jointly developing the RO-Crate specification and Open Source tools, as well as providing support and considering new use cases. The RO-Crate Community is open for anyone to join, to equally participate under a code of conduct, and currently has more than 40 members.

# RO-Crate Tooling

The work of the community led to the development of a number of tools for creating and using RO-Crates. Table shows the current set of implementations. Reviewing this list, one can see that there are tools that support commonly used programming languages including Python, Javascript, and Ruby. Additionally, these tools can integrate with commonly used research environments; in particular, the command line (*ro-crate-html-js*). Furthermore, there are tools that cater to the end user (*Describo, Workflow Hub*). For example, Describo was developed to help researchers of the Australian Criminal Characters project, in annotating historical prisoner records, to gain greater insight into the history of Australia [17].

While the development of these tools is promising, our analysis of their maturity status show that the majority are in the Beta stage. This is partly due to the fact that the RO-Crate specification itself only recently reached 1.0 status, in November 2019 [18]. Now that there is a fixed point of reference, and RO-Crate 1.1 (October 2020) [19] has stabilised, based on feedback from application development, we expect to see a further increase in the maturity of these tools.

Given the stage of the specification, the targets of these tools have been primarily developers, essentially providing the core libraries for working with RO-Crates. Another target has been that of research data managers who need to manage and curate large amounts of data.

We argue that the adoption of simple web technologies in the RO-Crate specification has led to the rapid development of this wide variety of tools.

| Tool Name | Targets | Language / Platform | Status | Brief Description |
|---|---|---|---|---|
| Describo [**???**] | Research Data Managers | NodeJS (Desktop) | RC | Interactive desktop application to create, update and export RO-Crates for different profiles |
| Describo Online [**???**] | Platform developers | NodeJS (Web) | Alpha | Web-based application to create RO-Crates using cloud storage |
| ro-crate-excel [**???**] | Data managers | JavaScript | Beta | Command-line tool to help create RO-Crates and HTML-readable rendering |

| Tool Name | Targets | Language / Platform | Status | Brief Description |
|---|---|---|---|---|
| ro-crate-html-js [???] | Developers | JavaScript | Beta | HTML rendering of RO-Crate |
| ro-crate-js [???] | Research Data Managers | JavaScript | Alpha | Library for creating/manipulating crates; basic validation code |
| ro-crate-ruby [???] | Developers | Ruby | Beta | Ruby library for reading/writing RO-Crate, with workflow support |
| ro-crate-py [???] | Developers | Python | Alpha | Object-oriented Python library for reading/writing RO-Crate |
| WorkflowHub [???] | Workflow users | Ruby | Beta | Workflow repository; imports and exports Workflow RO-Crate |
| Life Monitor [???] | Workflow developers | Python | Alpha | Workflow testing and monitoring service; Workflow Testing profile of RO-Crate |
| SCHeMa [20] | Workflow users | PHP | Alpha | Workflow execution using RO-Crate as exchange mechanism |
| galaxy2cwl [???] | Workflow developers | Python | Alpha | Wraps Galaxy workflow as Workflow RO-Crate |
| Modern PARADISEC [???] | Repository managers | Platform | Beta | Cultural Heritage portal based on OCFL and RO-Crate |
| ONI express [???] | Repository managers | Platform | Beta | Platform for publishing data and documents stored in an OCFL repository via a web interface |
| ocfl-tools [???] | Developers | JavaScript (CLI) | Beta | Tools for managing RO-Crates in an OCFL repository |
| RO Composer [???] | Repository developers | Java | Alpha | REST API for gradually building ROs for given profile. |
| Ro-Crate_2_ma-DMP [21] | Data Management Plan users | Python | Beta | Convert between machine-actionable data management plans (maDMP) and RO-Crate [22] |

**Table 1**: Applications and libraries implementing RO-Crate, targeting different types of users across multiple programming languages. Status is indicative as assessed by this work.

# Using RO-Crate

RO-Crate is fundamentally an infrastructure to help make FAIR research artefacts. In other words, the key question is whether RO-Crate can be used to share and (re)use research artefacts. Here we look at three research domains where it is being applied: Bioinformatics, Regulatory Science and Cultural Heritages. In addition we note how RO-Crate can have an importent role as part of machine-actionable data management plans.

## Bioinformatics workflows

WorkflowHub.eu is a European cross-domain registry of computational workflows, supported by European Open Science Cloud projects like EOSC-Life and research infrastructures such as the pan-

European bioinformatics network ELIXIR [16]. As part of promoting workflows as reusable tools, the WorkflowHub includes documentation and high-level rendering of the workflow structure independent of its native workflow definition format - the rationale is that a domain scientist looking for a particular computational analysis can browse all relevant workflows before narrowing down their workflow engine requirements. As such, the WorkflowHub is intended largely as a registry of workflows already deposited in repositories specific to particular workflow languages and domains, such as UseGalaxy.eu and Nextflow nf-core.

Being cross-domain also means the WorkflowHub has to cater for many different workflow systems. Many of these, for instance Nextflow and Snakemake, by virtue of their script-like nature, reference multiple neighbouring files typically maintained in a GitHub repository. WorkflowHub has therefore adapted RO-Crate as the packaging mechanism [23], by typing and annotating the constituent files of a workflow, crucially marking up the workflow language, as many workflow engines use common file extensions like `*.xml` and `*.json`. Workflows are further described with authors, license, diagram previews and a listing of their inputs and outputs. RO-Crates can thus be used for interoperable deposition of workflows to the WorkflowHub, but are also used as an archive for downloading workflows, embedding metadata registered with the WorkflowHub entry and translated workflow definitions such as abstract CWL and diagrams [24].

In WorkflowHub, RO-Crate therefore can be seen as taking on the role of an interoperability layer between registries, repositories and users. The iterative development between WorkflowHub developers and the RO-Crate community heavily informed the creation of the Bioschemas [**???**] profile for Computational Workflow, which again informed the RO-Crate 1.1 specification on workflows and led to the RO-Crate Python library [**???**] and WorkflowHub's Workflow RO-Crate profile, which, in a similar fashion to RO-Crate itself, recommends which workflow resources and descriptions are required. This co-development across project boundaries exemplifies the drive for simplicity and for establishing best practices.

While RO-Crates in WorkflowHub so far have been focused on workflows that are ready to be run, they are now moving into the development of a *Workflow Run RO-Crate profile* for the purposes of benchmarking, testing and executing workflows. As such, RO-Crate serves as container of both a *workflow definition* that may be executed and of a particular *workflow execution with test results*. This profile is a continuation of our previous work with capturing workflow provenance in a Research Object in CWLProv [25] and TavernaPROV [26], but rather than detailing every task execution in PROV with all intermediate data, this profile will focus on a schema.org provenance of the input/output boundary of the overall workflow execution. This *Level 1 workflow provenance* [25] can be expressed generally across workflow languages or engine types, with the option of more detailed provenance traces as separate resources in the RO-Crate.

WorkflowHub has recently enabled minting of DOIs for registered workflows, e.g. `10.48546/workflowhub.workflow.56.1` [27], lowering the barrier for citing workflows as computational methods along with their FAIR metadata – captured within an RO-Crate.

The value of computational workflows, however, is potentially undermined by the "collapse" over time of the software and services they depend upon: where software dependencies change in a non-backwards-compatible manner, or active maintenance ceases; an external resource, such as a reference index or a database query service, could shift to a different URL or modify its access protocol. In addition, the workflow itself may develop hard-to-find bugs as it is updated. This can take a big toll on the workflow's reusability and on the reproducibility of any processes it evokes.

For this reason, WorkflowHub is complemented by a monitoring and testing service called LifeMonitor[**???**], also supported by EOSC-Life. LifeMonitor's main goal is to assist in the creation, periodic execution and monitoring of workflow tests, enabling the early detection of software collapse

in order to minimise its detrimental effects. The communication of metadata related to workflow testing is achieved through the adoption of a [Workflow Testing RO-Crate profile](#) stacked on top of the *Workflow RO-Crate* profile. This further specialisation of Workflow RO-Crate allows to specify additional testing-related entities (test suites, instances, services, etc.), leveraging [RO-Crate's extension mechanism](#) through the addition of terms from custom namespaces.

In addition to showcasing RO-Crate's extensibility, the testing profile is an example of the format's flexibility and adaptability to the different needs of the research community. Though ultimately related to a computational workflow, in fact, most of the testing-specific entities are more about describing a protocol for interacting with a monitoring service than a set of research outputs and its associated metadata. Indeed, one of LifeMonitor's main functionalities is monitoring and reporting on test suites running on existing CI services, which is described in terms of service URLs and job identifiers in the testing profile. In principle, in this context, data could disappear altogether, leading to an RO-Crate consisting entirely of contextual entities. Such an RO-Crate acts more as an exchange format for communication between services (WorkflowHub and LifeMonitor) than as an aggregator for research data and metadata, providing a good example of the format's high versatility.

# Regulatory Sciences

[BioCompute Objects](#) (BCO) [28] is a community-led effort to standardise submissions of computational workflows to biomedical regulators. For instance, a genomics sequencing pipeline, as part of a personalised cancer treatment study, can be submitted to the US Food and Drugs Administration (FDA) for approval. BCOs are formalised in the standard IEEE 2791-2020 [29] as a combination of [JSON Schemas](#) that define the structure of JSON metadata files describing exemplar workflow runs in detail, covering aspects such as the usability and error domain of the workflow, its runtime requirements, the reference datasets used and representative output data produced.

While a BCO can give a structured view over a particular workflow, informing regulators about its workings independently of the underlying workflow definition language, it has only limited support for additional metadata. For instance, while the BCO itself can indicate authors and contributors, and in particular regulators and their review decisions, it cannot describe the provenance of individual data files or workflow definitions beyond assigning them global identifiers in the form of IRIs.

As a custom JSON format, BCOs cannot be extended with Linked Data concepts, except by adding an additional top-level JSON object formalised in another JSON Schema. A BCO and workflow submitted by upload to a regulator will also frequently consist of multiple cross-related files. Crucially, there is no way to tell whether a given `*.json` file is a BCO file, except by reading its content and check for its `spec_version`; yet many tools and workflow systems also use JSON files in their internal formats.

We can then consider how a BCO and its referenced artefacts can be packaged and transferred following FAIR principles. [BCO RO-Crate](#), part of the BioCompute Object user guides, defines a set of best practices for wrapping a BCO within a workflow, together with its exemplar outputs in an RO-Crate, which then provides typing and additional provenance metadata of the individual files, workflow definition, referenced data and the BCO metadata itself. While the BCO remains rigid, the RO-Crate is more open-ended and can therefore also describe other files in the submission not directly relatable from the BCO, such as further workflow documentation.

While there is some overlap in that RO-Crate can also describe a computational workflow, as detailed in previous sections, a *separation of concerns* emerges, where the BCO is responsible for describing the inside of a workflow and its run at an abstraction level suitable for a domain scientist, while the RO-Crate describes the surroundings of the workflow, classifying and relating its resources and providing provenance of their existence beyond the BCO.

A similar separation of concerns can be found within the RO-Crate itself, where the transport-level metadata, such as checksum of files, are [delegated to BagIt](#) manifests, a standard focusing on the preservation challenges of digital libraries[10]. As such RO-Crates are not required to list all the files in its folder hierarchy, only those that are deemed to be need a description.

Going deeper, a BCO alone is insufficient for reliable re-execution of a workflow, which would need a compatible workflow engine depending on the workflow definition language, so IEEE 2791 recommends using Common Workflow Language [30] for interoperable pipeline execution. CWL itself relies on tool packaging in software containers using Docker or Conda. As such we can consider BCO-RO-Crate as a stack consisting of transport-level manifest of files (BagIt), provenance, typing and context of those files (RO-Crate), workflow overview and purpose (BCO), interoperable workflow definition (CWL) and tool distribution (Docker).
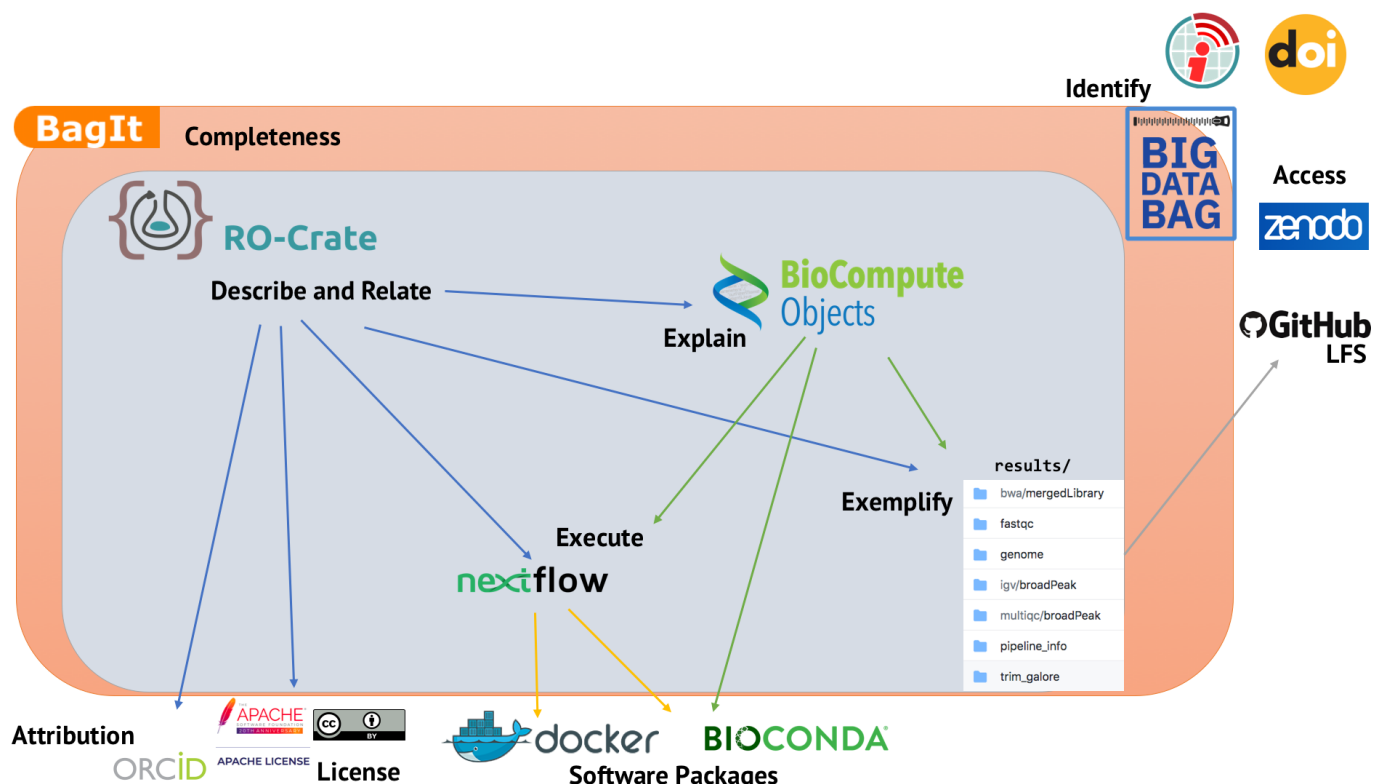


**Separation of Concerns in BCO RO-Crate**. BioCompute Object (IEEE2791) is a JSON file that structurally explains the purpose and implementation of a computational workflow, for instance implemented in Nextflow, that installs the workflow's software dependencies as Docker containers or BioConda packages. An example execution of the workflow shows the different kinds of result outputs, which may be external, using GitHub LFS to support larger data. The RO-Crate gathers all these local and external resources, relating them and giving individual descriptions, for instance permanent DOI identifiers for reused datasets accessed from Zenodo, but also adding external identifiers to attribute authors using ORCID or to identify which licenses apply to individual resources. The RO-Crate and its local files are captured in a BagIt whose checksum ensures completeness, combined with Big Data Bag [**31**]  features to "complete" the bag with large external files such as the workflow outputs

# Digital Humanities: Cultural Heritage

[PARADISEC](#) (the Pacific And Regional Archive for Digital Sources in Endangered Cultures) maintains a repository of more than 500,000 files documenting endangered languages across more than 16.000 items, collected over many years by researchers interviewing and recording native speakers across the region. As a proposed update of the 18 year old infrastructure, the [Modern PARADISEC demonstrator](#) has been [developed](#) to also help digitally preserve these artefacts using the [Oxford Common File Layout](#) (OCFL) for file consistency and RO-Crate for structuring and capturing the metadata of each item. The existing PARADISEC data collection has been ported and captured as RO-Crates. A Web portal then exposes the repository and its entries by indexing the RO-Crate metadata

files using Elasticsearch as a "NoSQL" object database, presenting a domain-specific view of the items - the RO-Crate is "hidden" and does not change the user interface.

This use case takes advantage of several RO-Crate features and principles. Firstly, the transcribed metadata is now independent of the PARADISEC platform and can be archived, preserved and processed in its own right, using schema.org vocabularies augmented with PARADISEC-specific terms. The lightweight infrastructure with RO-Crate as the holder of itemised metadata in regular files (organized using OCFL[**???**], with checksums for integrity checking and versioning) also gives flexibility for future developments and maintenance, for example, potentially using Linked Data software such as a graph database, queried using SPARQL triple patterns across RO-Crates, or a "last resort" fallback to the generic RO-Crate HTML preview[**???**], which can be hosted as static files by any web server, in line with the approach taken by the Endings Project[2].

## Machine-actionable Data Management Plans

Machine-actionable Data Management Plans (maDMPs) have been proposed as an improvement to automate FAIR data management tasks in research [32], e.g. by using PIDs and controlled vocabularies to describe what happens to data over the research lifecycle. The Research Data Alliance's *DMP Common Standard* for maDMPs [33] is one such formalisation for expressing maDMPs, which can be expressed as Linked Data using the DMP Common Standard Ontology [34], a specializaton W3C DCAT [**???**]. However, more commonly, RDA maDMPs are expressed using regular JSON, conforming to the DMP JSON Schema.

A mapping has been produced between Research Object Crates and such Machine-actionable Data Management Plans [22], implemented by the RO-Crate {RDA maDMP Mapper [35]. A similar mapping has been implemented by Ro-Crate_2_ma-DMP [21]. In both cases, a maDMP can be converted to a RO-Crate, or vice versa. In [22] this functionality caters for two use cases:

1. Start a skeleton data management plan based on an existing RO-Crate dataset, e.g. from an RO-Crate from Workflow Hub
2. Instantiate an RO-Crate based on a data management plan.

An important difference here is that data management plans are (ideally) written in advance of data production, while RO-Crates are typically created to describe data after it has been generated. This approach shows the importance of *templating* to make both tasks more automatable and achievable, and how RO-Crate can fit into earlier stages of the research life cycle.

## Institutional data repositories – Harvard Data Commons

# Related Work

With the increasing digitization of research processes, there has been a significant call for the wider use of interoperable sharing of data and its associated metadata. For a comprehensive overview of this literature and recommendations in particular for data, we refer to [36], which highlights the wide variety of metadata and documentation that the literature prescribes for enabling the reuse of data.

Here, instead of surveying the large literature on sharing digital scholarly artefacts, we rather focus on approaches to bundling such artefacts along with their metadata. This notion has a long history [37], but recent approaches have followed three strands: 1) publishing to centralised repositories; 2) packaging approaches similar to RO-Crate; and 3) bundling the computational workflow around a scientific experiment.

# Bundling and Packaging Digital Research Artefacts

The challenge of describing computational workflows was one of the main motivations for the early proposal of *Research Objects*[8] as first-class citizens for sharing and publishing, by bundling datasets, workflows, scripts and results along with traditional dissemination materials like journal articles and presentations, forming a single package. Crucially, these resources are not just gathered, but also individually typed, described and related to each-other using semantic vocabularies. As pointed out in [8] an open-ended *Linked Data* approach is not sufficient for scholarly communication, as a common data model is also needed in addition to common practices for managing and annotating lifecycle, ownership, versioning and attributions.

Considering the FAIR principles, we can say with hindsight that the initial Research Objects approaches strongly targeted *Interoperability*, with a particular focus on reproducibility with computational workflows and reuse of existing RDF vocabularies.

The first implementation of Research Objects in 2009 for sharing workflows in myExperiment [38] was based on RDF ontologies [**???**], building on Dublin Core, FOAF, SIOC, Creative Commons, OAI-ORE and (later) DBPedia to form myExperiment ontologies for describing social networking, attribution and credit, annotations, aggregation packs, experiments, view statistics, contributions, and workflow components. [**???**]

Programmatic access to Research Objects was facilitated with an RDF endpoint that exposed individual myExperiment resources, also queryable from a SPARQL endpoint, both using the myExperiment vocabularies and RDF formats RDF/XML and Turtle.

# FAIR Digital Objects

FAIR Digital Objects (FDO) [39] have been proposed as a conceptual framework for making digital resources available in a Digital Objects (DO) architecture that encourages active use of the objects and their metadata. In particular, an FDO has five parts: (i) The FDO *content*, bit sequences stored in an accessible repository; (ii) a *Permanent Identifier* (PID) such as a DOI that identifies the FDO and can resolve these parts; (iii) Associated rich *metadata*, as separate FDOs; (iv) Type definitions, also separate FDOs; (v) Associated *operations* for the given types. A Digital Object typed as a Collection aggregates other DOs by reference.

As an "abstract protocol", Digital Objects could be implemented in multiple ways. One suggested implementation is the FAIR Digital Object Framework, based on HTTP and Linked Data. While there is agreement on using permanent identifiers based on DOI, consensus on how to represent common metadata, core types and collections as FDOs has not yet been reached. We argue that RO-Crate can play an important role for FDOs:

1. By providing a predictable and extensible serialization of structured metadata
2. By formalizing how to aggregate digital objects as a collection (and adding their context)
3. By providing a natural Metadata FDO in the form of the RO-Crate Metadata File
4. By being based on Linked Data and schema.org vocabulary, meaning that PIDs already exist for common types and properties.

At the same time it is clear that the goal of FDO is broader than that of RO-Crate; namely FDOs are active objects with distributed operations, and add further constraints such as permanent identifiers for every element. While these measures can be said to improve FAIR features of digital objects and are also useful for RO-Crate, they also severely restrict the possible FDO infrastructure that needs to

be implemented and maintained in order for FDOs to remain available. RO-Crate, on the other hand, is more flexible: it can minimally be used within any file system structure, or ideally exposed through a range of Web-based scenarios, but a *FAIR profile of RO-Crate* (e.g. enforcing PID usage) will also fit well within, and help to implement, a FAIR Digital Object ecosystem.

# Packaging Workflows

The use of computational workflows has gained prominence, in particular in the life sciences, typically combining a chain of open source tools in an analytical pipeline. While workflows may have initially been used to improve scalability, it can be argued that they also assist in making computed data results FAIR. At the same time, however, they raise additional FAIR challenges, since they can be considered as important research artefacts themselves, posing the problem of capturing and explaining the computational methods behind the analysis [40].

Even when researchers follow current best practice for workflow reproducibility, [41] [42] the communication of outcomes through traditional academic publishing routes with a textual representation adds barriers that hinder reproducibility and FAIR use of the knowledge previously captured in the workflow.

As a real-life example, let's look at a metagenomics article [43] where the authors have gone to extraordinary efforts to document the individual tools that have been reused, including their citations, versions, settings, parameters and combinations. The *Methods* section is 2 pages in tight double-columns with 24 additional references, supported by data availability on an FTP server (60 GB) [**???**] and the open source code available in GitHub [Finn-Lab/MGS-gut](Finn-Lab/MGS-gut) [**???**] including the pipeline as shell scripts, and associated analysis scripts in R and Python.

This attention to reporting detail for computational workflows is unfortunately not yet the norm, and although bioinformatics journals have strong *data availability* requirements, they frequently do not require authors to include or cite *software, scripts and pipelines* used for analysing and producing results [**???**] – rather, authors might be penalised for doing so [cite?] as it would detrimentally count against arbitrary limits on number of pages and references.

However detailed this additional information might be, another researcher who wants to reuse a particular computational method may first want to assess if the described tool and workflow is Re-runnable (executable at all), Repeatable (same results for original inputs on same platform), Reproducible (same results for original inputs with different platform or newer tools) and ultimately Reusable (similar results for different input data), Repurposable (reusing parts of the method for making a new method) or Replicable (rewriting the workflow following the method description). [44] [**???**]

Following the textual description alone, researchers would be forced to jump straight to evaluate "Replicable" by rewriting the pipeline from scratch. This can be expensive and error-prone. They would firstly need to install all the software dependencies and reference datasets. This can be a daunting task in and of itself, which may have to be repeated multiple times as workflows typically are developed at small scale on desktop computers, scaled up to local clusters, and potentially productionized using cloud instances, each of which will have different requirements for software installations.

In recent years the situation has been greatly improved by software packaging and container technologies like Docker and Conda, which have seen increased adaptation in life sciences [45], with supporting collaborative efforts like BioConda [46], BioContainers [47] and by Linux distributions themselves (e.g. Debian Med [48]), to make more than 7000 software packages available [in BioConda

alone] (https://anaconda.org/bioconda/) and 9000 containers [in BioContainers](). Docker and Conda have gained integration in workflow systems like Snakemake, Galaxy, Nextflow, meaning a downloaded workflow definition can now be executed on a "blank" machine (except for the workflow engine) with the underlying analytical tools installed on demand.

# Conclusion

RO-Crate provides a lightweight approach to packaging digital research artefacts with structured metadata, assisting developers and researchers to produce and consume FAIR data archives of their Research Objects, including permanent identifiers, contextualization and provenance of research items. Aggregated data may be large and distributed, or regular folders on a file system.

As a set of Best Practice recommendations, developed by an open and broad community, RO-Crate shows how to use "just enough" Linked Data standards in a consistent way, with structured metadata using a rich base vocabulary that can cover everyday contextual relations, whilst retaining extensiblility to domain- and application-specific uses.

Supported by multiple open source tools and libraries, RO-Crate fits into the larger landscape of open scholarly communication and FAIR Digital Object infrastructure, being easily integrated into data repository platforms. RO-Crate can be applied as a data/metadata exchange mechanism, assist in long-term archival preservation of metadata and data, or simply used at small-scale by individual researchers.

# Acknowledgements

## Contributions

Author contributions according to the Contributor Roles Taxonomy [CASRAI CrEDiT]() [[49]]():

- **Stian Soiland-Reyes**: Conceptualization, Data curation, Formal Analysis, Funding acquisition, Investigation, Methodology, Project administration, Software, Visualization, Writing – original draft, Writing – review & editing
- **Peter Sefton**: Conceptualization, Methodology, Project administration, Software, Writing – review & editing
- **Simone Leo**: Software, Writing – review & editing
- **Ana Trisovic**: Software, Writing – review & editing
- **RO-Crate Community**: Software, Validation, Writing – review & editing
- **Paul Groth**: Methodology, Supervision, Writing – original draft, Writing – review & editing
- **Carole Goble**: Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Visualization, Writing – review & editing

We would also like to thank:

- **Mercè Crosas**: Writing – review & editing
- **Daniel Garijo**: Methodology

- **Eoghan Ó Carragáin**: Project administration, Writing – review & editing
- **Finn Bacall**: Software, Methodology
- **Frederik Coppens**: Writing – review & editing
- **Herbert Van de Sompel**: Writing – review & editing
- **Ignacio Eguinoa**: Software
- **José María Fernández**: Methodology
- **Marc Portier**: Methodology
- **Marco La Rosa**: Software, Methodology
- **Nick Juty**: Writing – review & editing
- **Oscar Corcho**: Writing – review & editing
- **Stuart Owen**: Writing – review & editing

# Formalizing RO-Crate in First Order Logic

Below is an attempt to formalize the concept of RO-Crate as a set of relations using First Order Logic:

## Language

```
𝕃ro-crate = { Property(p), Class(c), Literal(x), ℝ, 𝕊 }
𝔻 =  𝕀ri
𝕀ri ≡  { IRIs as defined in RFC3987 }
ℝ ≡  { real or integer numbers }
𝕊 ≡  { literal strings }
```

The domain of discourse is the set of `𝕀ri` identifiers [12] (notation `<http://example.com/>`), with additional descriptions using numbers `ℝ` (notation `13.37`) and literal strings `𝕊` (notation `"Hello"`).

From this formalized language `𝕃ro-crate` an RO-Crate can be interpreted in any representation that can gather these descriptions, their properties, classes, and literal attributes.

## Minimal RO-Crate

Below is using `𝕃ro-crate` to define a minimal RO-Crate:

```
RO-Crate(R) ⊨  Root(R) ∧ Mentions(R, R) ∧
    hasPart(R, d) ∧ Mentions(R, d) ∧ DataEntity(d) ∧
    Mentions(R, c) ∧ ContextualEntity(c)
∀r Root(r) →  Dataset(r) ∧ name(r, n)∧ description(r, d)
                        ∧ published(r, date) ∧ license(e, l)
∀e∀n name(e, n) →  Literal(n)
∀e∀d description(e, d) →  Literal(d)
∀e∀date datePublished(e, date) →  Literal(date)
∀e∀l license(e, l) →  ContextualEntity(l)
DataEntity(e) ≡  File(e) ⊕ Dataset(e)
Entity(e) ≡  DataEntity(e) ∨ ContextualEntity(e)
∀e Entity(e) → Class(e)
Mentions(R, s) ⊨  Relation(s, p, e)  ⊕  Attribute(s, p, l)
Relation(s, p, o) ⊨ Entity(s) ∧ Property(p) ∧ Entity(o)
Attribute(s, p, x) ⊨ Entity(s)  ∧ Property(p) ∧ Literal(x)
Literal(x) ≡  x ∈ ℝ  ⊕   x ∈ 𝕊
```

An `RO-Crate(R)` is defined as a self-described *Root Data Entity*, which describes and contains parts (*data entities*), which are further described in *contextual entities*. These terms align with their use in the [RO-Crate 1.1 terminology](.).

The `Root(r)` is a type of `Dataset(r)`, and must have the metadata to literal attributes to provide a `name`, `description` and `datePublished`, as well as a contextual entity identifying its license. These predicates correspond to the RO-Crate 1.1 [requirements for the root data entity](.).

The concept of an `Entity(e)` is introduced as being either a DataEntity(e), a `ContextualEntity(e)`, or [both](.); and must be typed with at least one `Class(e)`.

For simplicity in this formalization (and to assist production rules below) `R` is a constant representing a single RO-Crate, typically written to independent RO-Crate Metadata files. `R` is used by `Mentions(R, e)` to indicate that `e` is an Entity described by the RO-Crate and therefore its metadata (a set of Relation and Attribute predicates) form part of the RO-Crate serialization. `Relation(s, p, o)` and `Attribute(s, p, x)` are defined as a *subject-predicate-object* triple pattern from an `Entity(s)` using a `Property(p)` to either another `Entity(o)` or a `Literal(x)` value.

## Example of formalized RO-Crate

The below is an example RO-Crate represented using the above formalization, assuming a base URI of `http://example.com/ro/123/`:

```
RO-Crate(<http://example.com/ro/123/>)
name(<http://example.com/ro/123/,
    "Data files associated with the manuscript:Effects of …")
description(<http://example.com/ro/123/,
    "Palliative care planning for nursing home residents …")
datePublished(<http://example.com/ro/123/>, "2017")
license(<http://example.com/ro/123/>,
    <https://creativecommons.org/licenses/by-nc-sa/3.0/au/>
ContextualEntity(<https://creativecommons.org/licenses/by-nc-sa/3.0/au/>)
name(<https://creativecommons.org/licenses/by-nc-sa/3.0/au/,
    "Attribution-NonCommercial-ShareAlike 3.0 Australia (CC BY-NC-SA 3.0
AU)")

hasPart(<http://example.com/ro/123/>, <http://example.com/ro/123/file.txt>)
File(<http://example.com/ro/123/survey.csv>)
name(<http://example.com/ro/123/survey.csv>, "Survey of care providers")
hasPart(<http://example.com/ro/123/>,
<http://www.example.om/ro/123/folder/>)
Dataset(<http://example.com/ro/123/interviews/>)
name(<http://example.com/ro/123/interviews/>,
    "Audio recordings of care provider interviews")
```

In reality many additional attributes from schema.org types like http://schema.org/Dataset and http://schema.org/CreativeWork would be used to further describe the RO-Crate and its entities, but as these are optional they do not form part of this formalization.

## Mapping to RDF with schema.org

A formalized RO-Crate can be mapped to different serializations. Below follows a mapping to RDF using schema.org.

```
Dataset(d) →  type(d, <http://schema.org/Dataset>)
File(f) →  type(f, <http://schema.org/MediaObject>)
Property(p) →  type(p, <http://www.w3.org/2000/01/rdf-schema#Property>)
Class(c) →  type(c, <http://www.w3.org/2000/01/rdf-schema#Class>)
CreativeWork(e) →  ContextualEntity(e) ∧ type(e,
<http://schema.org/CreativeWork>)
hasPart(e, t) →  Relation(e, <http://schema.org/hasPart>, t)
type(e, t) →  Relation(e, <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,
t) ∧ Class(t)
name(e, n) →  Attribute(e, <http://schema.org/name>, n)
description(e, d) →  Attribute(e, <http://schema.org/description>, d)
datePublished(e, date) →  Attribute(e, <http://schema.org/datePublished>,
date)
license(e, l) →  Relation(e, <http://schema.org/license>, l) ∧
CreativeWork(l)
```

Note that in the JSON-LD serialization of RO-Crate the expression of `Class` and `Property` is typically indirect, as the JSON-LD `@context` maps to schema.org IRIs, which when resolved as Linked Data embeds their formal definition as RDFa.

## RO-Crate 1.1 Metadata File Descriptor

An important RO-Crate principle is that of being **self-describing**. Therefore the serialization of the RO-Crate into a file should also describe itself in a [Metadata File Descriptor](), indicating it is about (describing) the RO-Crate root data entity, and that it conformsTo a particular version of the RO-Crate specification:

```
about(s,o) →  Relation(s, <http://schema.org/about>, o)
conformsTo(s,o) →  Relation(s, <http://purl.org/dc/terms/conformsTo>, R)
MetadataFileDescriptor(m) →  ( CreativeWork(m) ∧ about(m,R) ∧ RO-Crate(R) ∧
     conformsTo(m, <https://w3id.org/ro/crate/1.1>) )
```

Note that although the metadata file necessarily is an *information resource* written to disk or served over the network (e.g. as JSON-LD), it is not considered to be a contained *part* of the RO-Crate in the form of a *data entity*, rather it is described only as a *contextual entity*.

While in the conceptual model the *RO-Crate Metadata File* can be seen as the top-level node that describes the *RO-Crate Root*, in the formal model (and the JSON-LD format) the metadata file descriptor is an additional contextual entity and not affecting the depth-limit of the RO-Crate.

## Forward-chained Production Rules for JSON-LD

Combining the above predicates and schema.org mapping with rudimentary JSON templates, these forward-chaining production rules can output JSON-LD according to the RO-Crate 1.1 specification[3]:

```
Mentions(R, s) ∧ Relation(s, p, o) →  Mentions(R, o)
i ∈ 𝕀ri → "i"
r ∈ ℝ →  r
s ∈ 𝕊 → "s"
∀s∀p∀o Relation(s,p,o) →  { "@id": s,
                            p: { "@id": o }
                          }
∀s∀p∀v Attribute(s,p,v) →  { "@id": s,
                            p: v
                          }
∀r∀c  RO-Crate(r) →  { "@graph": [ Mentions(r, c)* ] }
R ⊨  <./>
MetadataFileDescriptor(<ro-crate-metadata.json>)
```

This exposes the first order logic domain of discourse of IRIs, with rational numbers and strings as their corresponding JSON-LD representation. These production rules first grow the graph of `R` by adding a transitive rule that anything described in `R` which is related to `o` means that `o` is also mentioned by the RO-Crate `R`. For simplicity this rule is one-way; in practice the JSON-LD graph can

also contain free-standing contextual entities that have outgoing relations to data- and contextual entities.

# RO-Crate Community

As of 2021-05-07, the *RO-Crate* Community members are:

- Peter Sefton (co-chair)
- Stian Soiland-Reyes (co-chair)
- Eoghan Ó Carragáin (emeritus chair)
- Oscar Corcho
- Daniel Garijo
- Raul Palma
- Frederik Coppens
- Carole Goble
- José María Fernández
- Kyle Chard
- Jose Manuel Gomez-Perez
- Michael R Crusoe
- Ignacio Eguinoa
- Nick Juty
- Kristi Holmes
- Jason A. Clark
- Salvador Capella-Gutierrez
- Alasdair J. G. Gray
- Stuart Owen
- Alan R Williams
- Giacomo Tartari
- Finn Bacall
- Thomas Thelen
- Hervé Ménager
- Laura Rodríguez-Navas
- Paul Walk
- brandon whitehead
- Mark Wilkinson
- Paul Groth
- Erich Bremer
- LJ Garcia Castro
- Karl Sebby
- Alexander Kanitz
- Ana Trisovic
- Gavin Kennedy
- Mark Graves
- Jasper Koehorst
- Simone Leo
- Marc Portier
- Paul Brack
- Milan Ojsteršek
- Bert Droesbeke
- Chenxu Niu
- Kosuke Tanabe

# References

1. **Enhancing reproducibility for computational methods**
   V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. A. Ioannidis, M. Taufer
   *Science* (2016-12-08) https://doi.org/gbr42b
   DOI: 10.1126/science.aah6168 · PMID: 27940837

2. **Towards FAIR principles for research software**
   Anna-Lena Lamprecht, Leyla Garcia, Mateusz Kuzak, Carlos Martinez, Ricardo Arcila, Eva Martin Del Pico, Victoria Dominguez Del Angel, Stephanie van de Sandt, Jon Ison, Paula Andrea Martinez, … Salvador Capella-Gutierrez
   *Data Science* (2020-06-12) https://doi.org/gg66q6
   DOI: 10.3233/ds-190026

3. **The FAIR Guiding Principles for scientific data management and stewardship**
   Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, … Barend Mons
   *Scientific Data* (2016-03-15) https://doi.org/bdd4
   DOI: 10.1038/sdata.2016.18 · PMID: 26978244 · PMCID: PMC4792175

4. **Data stewardship for discovery: implementing fair principles**
   Barend Mons
   (2018) http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5323689
   ISBN: 9781315351148

5. **Zenodo, an Archive and Publishing Repository: A tale of two herbarium specimen pilot projects**
   Mathias Dillen, Quentin Groom, Donat Agosti, Lars Nielsen
   *Biodiversity Information Science and Standards* (2019-06-18) https://doi.org/ghk2kn
   DOI: 10.3897/biss.3.37080

6. **The worldwide Protein Data Bank (wwPDB): ensuring a single, uniform archive of PDB data**
   H. Berman, K. Henrick, H. Nakamura, J. L. Markley
   *Nucleic Acids Research* (2007-01-03) https://doi.org/bv4qhz
   DOI: 10.1093/nar/gkl971 · PMID: 17142228 · PMCID: PMC1669775

7. **Ten Simple Rules for Reproducible Computational Research**
   Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, Eivind Hovig
   *PLoS Computational Biology* (2013-10-24) https://doi.org/pjb
   DOI: 10.1371/journal.pcbi.1003285 · PMID: 24204232 · PMCID: PMC3812051

8. **Why linked data is not enough for scientists**
   Sean Bechhofer, Iain Buchan, David De Roure, Paolo Missier, John Ainsworth, Jiten Bhagat, Philip Couch, Don Cruickshank, Mark Delderfield, Ian Dunlop, … Carole Goble
   *Future Generation Computer Systems* (2013-02) https://doi.org/bgmqrb
   DOI: 10.1016/j.future.2011.08.004

9. **A lightweight approach to research object data packaging**
   Eoghan Ó Carragáin, Carole Goble, Peter Sefton, Stian Soiland-Reyes

*Zenodo* (2019-06-20) https://doi.org/gf5twv
DOI: 10.5281/zenodo.3250687

10. **The BagIt File Packaging Format (V1.0)**
J. Kunze, J. Littman, E. Madden, J. Scancella, C. Adams
*RFC Editor* (2018-10) https://doi.org/gjvnbj
DOI: 10.17487/rfc8493

11. **Linked Data**
Christian Bizer, Tom Heath, Tim Berners-Lee
*IGI Global* (2011) https://doi.org/dh8v52
DOI: 10.4018/978-1-60960-593-3.ch008

12. **Internationalized Resource Identifiers (IRIs)**
M. Duerst, M. Suignard
*RFC Editor* (2005-01) https://doi.org/gjvnbg
DOI: 10.17487/rfc3987

13. **RO-Crate Metadata Specification 1.1.1**
Peter Sefton, Eoghan Ó Carragáin, Stian Soiland-Reyes, Oscar Corcho, Daniel Garijo, Raul Palma, Frederik Coppens, Carole Goble, José María Fernández, Kyle Chard, … Marc Portier
*Zenodo* (2021-02-15) https://doi.org/gjvnbk
DOI: 10.5281/zenodo.4541002

14. **Using a suite of ontologies for preserving workflow-centric research objects**
Khalid Belhajjame, Jun Zhao, Daniel Garijo, Matthew Gamble, Kristina Hettne, Raul Palma, Eleni Mina, Oscar Corcho, José Manuel Gómez-Pérez, Sean Bechhofer, … Carole Goble
*Journal of Web Semantics* (2015-05) https://doi.org/f73k3h
DOI: 10.1016/j.websem.2015.01.003

15. **Datacrate Submisssion To The Workshop On Research Objects**
Peter Sefton
*Zenodo* (2018-07-15) https://doi.org/gf5twt
DOI: 10.5281/zenodo.1445817

16. **ELIXIR: a distributed infrastructure for European biological data**
Lindsey C. Crosswell, Janet M. Thornton
*Trends in Biotechnology* (2012-05) https://doi.org/gjvnpq
DOI: 10.1016/j.tibtech.2012.02.002 · PMID: 22417641

17. **Digital crowdsourcing and public understandings of the past: citizen historians meet Criminal Characters**
Alana Piper
*History Australia* (2020-08-14) https://doi.org/gjvnbv
DOI: 10.1080/14490854.2020.1796500

18. **RO-Crate Metadata Specification 1.0**
Peter Sefton, Eoghan Ó Carragáin, Stian Soiland-Reyes, Oscar Corcho, Daniel Garijo, Raul Palma, Frederik Coppens, Carole Goble, José María Fernández, Kyle Chard, … Thomas Thelen
*Zenodo* (2019-11-15) https://doi.org/gjxkh7
DOI: 10.5281/zenodo.3541888

19. **RO-Crate Metadata Specification 1.1**
Peter Sefton, Eoghan Ó Carragáin, Stian Soiland-Reyes, Oscar Corcho, Daniel Garijo, Raul Palma, Frederik Coppens, Carole Goble, José María Fernández, Kyle Chard, … Simone Leo
*Zenodo* (2020-10-30) https://doi.org/gjxkh8
DOI: 10.5281/zenodo.4031327

20. **SCHeMa: Scheduling Scientific Containers on a Cluster of Heterogeneous Machines**
Thanasis Vergoulis, Konstantinos Zagganas, Loukas Kavouras, Martin Reczko, Stelios Sartzetakis, Theodore Dalamagas
*arXiv* (2021-03-24) https://arxiv.org/abs/2103.13138v1

21. **BrennerG/Ro-Crate_2_ma-DMP: v1.0.0**
, BrennerG
*Zenodo* (2020-06-22) https://doi.org/gjx263
DOI: 10.5281/zenodo.3903463

22. **Research Object Crates and Machine-actionable Data Management Plans**
Tomasz Miksa, Maroua Jaoua, Ghaith Arfaoui
*PUBLISSO* (2020) https://doi.org/gjx26z
DOI: 10.4126/frl01-006423291

23. **EOSC-Life Methodology framework to enhance reproducibility within EOSC-Life**
Florence Bietrix, José Maria Carazo, Salvador Capella-Gutierrez, Frederik Coppens, Maria Luisa Chiusano, Romain David, Jose Maria Fernandez, Maddalena Fratelli, Jean-Karim Heriche, Carole Goble, … Jing Tang
*Zenodo* (2021-04-30) https://doi.org/gjx25c
DOI: 10.5281/zenodo.4705078

24. **Implementing FAIR Digital Objects in the EOSC-Life Workflow Collaboratory**
Carole Goble, Stian Soiland-Reyes, Finn Bacall, Stuart Owen, Alan Williams, Ignacio Eguinoa, Bert Droesbeke, Simone Leo, Luca Pireddu, Laura Rodríguez-Navas, … Frederik Coppens
*Zenodo* (2021-03-12) https://doi.org/gjvnpr
DOI: 10.5281/zenodo.4605654

25. **Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv**
Farah Zaib Khan, Stian Soiland-Reyes, Richard O Sinnott, Andrew Lonie, Carole Goble, Michael R Crusoe
*GigaScience* (2019-11) https://doi.org/gjx247
DOI: 10.1093/gigascience/giz095 · PMID: 31675414 · PMCID: PMC6824458

26. **Tracking Workflow Execution With Tavernaprov**
Stian Soiland-Reyes, Pinar Alper, Carole Goble
*Zenodo* (2016-05-11) https://doi.org/gjx25d
DOI: 10.5281/zenodo.51314

27. **Protein Ligand Complex MD Setup tutorial using BioExcel Building Blocks (biobb) (jupyter notebook)**
Douglas Lowe, Genís Bayarri
*WorkflowHub* (2021) https://doi.org/gjxkh6
DOI: 10.48546/workflowhub.workflow.56.1

28. **Enabling precision medicine via standard communication of HTS provenance, analysis, and results**
Gil Alterovitz, Dennis Dean, Carole Goble, Michael R. Crusoe, Stian Soiland-Reyes, Amanda Bell, Anais Hayes, Anita Suresh, Anjan Purkayastha, Charles H. King, … Raja Mazumder
*PLOS Biology* (2018-12-31) https://doi.org/gjvnbf
DOI: 10.1371/journal.pbio.3000099 · PMID: 30596645 · PMCID: PMC6338479

29. **:(unav)**
Institute of Electrical and Electronics Engineers (IEEE)
(2020-05-15) https://doi.org/gjvnbc
DOI: 10.1109/ieeestd.2020.9094416

30. **Common Workflow Language, v1.0**
Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, … Luka Stojanovic
*figshare* (2016) https://doi.org/gf6ppg
DOI: 10.6084/m9.figshare.3115156.v2

31. **I'll take that to go: Big data bags and minimal identifiers for exchange of large, complex datasets**
Kyle Chard, Mike D'Arcy, Ben Heavner, Ian Foster, Carl Kesselman, Ravi Madduri, Alexis Rodriguez, Stian Soiland-Reyes, Carole Goble, Kristi Clark, … Arthur Toga
*Institute of Electrical and Electronics Engineers (IEEE)* (2016-12) https://doi.org/gjxkh5
DOI: 10.1109/bigdata.2016.7840618

32. **Ten principles for machine-actionable data management plans**
Tomasz Miksa, Stephanie Simms, Daniel Mietchen, Sarah Jones
*PLOS Computational Biology* (2019-03-28) https://doi.org/ggbhsk
DOI: 10.1371/journal.pcbi.1006750 · PMID: 30921316 · PMCID: PMC6438441

33. **RDA DMP Common Standard for Machine-actionable Data Management Plans**
Paul Walk, Tomasz Miksa, Peter Neish
*Research Data Alliance* (2019) https://doi.org/ggd4qw
DOI: 10.15497/rda00039

34. **New version of the DMP Common Standard Ontology**
João Cardoso, Fajar J. Ekaputra, Leyla Garcia, Christine Jacquemot
*Zenodo* (2020-07-14) https://doi.org/gjx3ch
DOI: 10.5281/zenodo.3944464

35. **RO-Crate RDA maDMP Mapper**
Ghaith Arfaoui, Maroua Jaoua
*Zenodo* (2020-06-29) https://doi.org/gjx3cg
DOI: 10.5281/zenodo.3922136

36. **Dataset Reuse: Toward Translating Principles to Practice**
Laura Koesten, Pavlos Vougiouklis, Elena Simperl, Paul Groth
*Patterns* (2020-11) https://doi.org/ghqhvm
DOI: 10.1016/j.patter.2020.100136 · PMID: 33294873 · PMCID: PMC7691392

37. **Electronic documents give reproducible research a new meaning**
Jon F. Claerbout, Martin Karrenbach

*Society of Exploration Geophysicists* (1992-01) https://doi.org/b6t7wj
DOI: 10.1190/1.1822162

38. **myExperiment: a repository and social network for the sharing of bioinformatics workflows**
Carole A. Goble, Jiten Bhagat, Sergejs Aleksejevs, Don Cruickshank, Danius Michaelides, David Newman, Mark Borkum, Sean Bechhofer, Marco Roos, Peter Li, David De Roure
*Nucleic Acids Research* (2010-07-01) https://doi.org/cpkvqd
DOI: 10.1093/nar/gkq429 · PMID: 20501605 · PMCID: PMC2896080

39. **FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units**
Koenraad De Smedt, Dimitris Koureas, Peter Wittenburg
*Publications* (2020-04-11) https://doi.org/ggrw3w
DOI: 10.3390/publications8020021

40. **FAIR Computational Workflows**
Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R. Crusoe, Kristian Peters, Daniel Schober
*Data Intelligence* (2020-01) https://doi.org/gjvnhd
DOI: 10.1162/dint_a_00033

41. **Practical Computational Reproducibility in the Life Sciences**
Björn Grüning, John Chilton, Johannes Köster, Ryan Dale, Nicola Soranzo, Marius van den Beek, Jeremy Goecks, Rolf Backofen, Anton Nekrutenko, James Taylor
*Cell Systems* (2018-06) https://doi.org/ggdv3z
DOI: 10.1016/j.cels.2018.03.014 · PMID: 29953862 · PMCID: PMC6263957

42. **Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities**
Sarah Cohen-Boulakia, Khalid Belhajjame, Olivier Collin, Jérôme Chopard, Christine Froidevaux, Alban Gaignard, Konrad Hinsen, Pierre Larmande, Yvan Le Bras, Frédéric Lemoine, … Christophe Blanchet
*Future Generation Computer Systems* (2017-10) https://doi.org/ggb87f
DOI: 10.1016/j.future.2017.01.012

43. **A new genomic blueprint of the human gut microbiota**
Alexandre Almeida, Alex L. Mitchell, Miguel Boland, Samuel C. Forster, Gregory B. Gloor, Aleksandra Tarkowska, Trevor D. Lawley, Robert D. Finn
*Nature* (2019-02-11) https://doi.org/gfvfd8
DOI: 10.1038/s41586-019-0965-1 · PMID: 30745586 · PMCID: PMC6784870

44. **Re-run, Repeat, Reproduce, Reuse, Replicate: Transforming Code into Scientific Contributions**
Fabien C. Y. Benureau, Nicolas P. Rougier
*Frontiers in Neuroinformatics* (2018-01-04) https://doi.org/ggb79t
DOI: 10.3389/fninf.2017.00069 · PMID: 29354046 · PMCID: PMC5758530

45. **Robust Cross-Platform Workflows: How Technical and Scientific Communities Collaborate to Develop, Test and Share Best Practices for Data Analysis**
Steffen Möller, Stuart W. Prescott, Lars Wirzenius, Petter Reinholdtsen, Brad Chapman, Pjotr Prins, Stian Soiland-Reyes, Fabian Klötzl, Andrea Bagnacani, Matúš Kalaš, … Michael R. Crusoe
*Data Science and Engineering* (2017-11-16) https://doi.org/ggwrrk
DOI: 10.1007/s41019-017-0050-4

46. **Bioconda: sustainable and comprehensive software distribution for the life sciences**
Björn Grüning, Ryan Dale, Andreas Sjödin, Brad A. Chapman, Jillian Rowe, Christopher H. Tomkins-Tinch, Renan Valieris, Johannes Köster, The Bioconda Team
*Nature Methods* (2018-07-02) https://doi.org/gd2xzp
DOI: 10.1038/s41592-018-0046-7 · PMID: 29967506

47. **BioContainers: an open-source and community-driven framework for software standardization**
Felipe da Veiga Leprevost, Björn A Grüning, Saulo Alves Aflitos, Hannes L Röst, Julian Uszkoreit, Harald Barsnes, Marc Vaudel, Pablo Moreno, Laurent Gatto, Jonas Weber, … Yasset Perez-Riverol
*Bioinformatics* (2017-08-15) https://doi.org/gh9qqj
DOI: 10.1093/bioinformatics/btx192 · PMID: 28379341 · PMCID: PMC5870671

48. **Community-driven computational biology with Debian Linux**
Steffen Möller, Hajo Nils Krabbenhöft, Andreas Tille, David Paleino, Alan Williams, Katy Wolstencroft, Carole Goble, Richard Holland, Dominique Belhachemi, Charles Plessy
*BMC Bioinformatics* (2010-12-21) https://doi.org/bjxjw2
DOI: 10.1186/1471-2105-11-s12-s5 · PMID: 21210984 · PMCID: PMC3040531

49. **Beyond authorship: attribution, contribution, collaboration, and credit**
Amy Brand, Liz Allen, Micah Altman, Marjorie Hlava, Jo Scott
*Learned Publishing* (2015-04-01) https://doi.org/gc6v3m
DOI: 10.1087/20150211

---

1. IRIs[12] are a generalisation of URIs (which include well-known http/https URLs), permitting international Unicode characters without `%`-encoding, commonly used on the browser address bar and in HTML5.↩

2. The Endings Project is a five-year project funded by the Social Sciences and Humanities Research Council (SSHRC) that is creating tools, principles, policies and recommendations for digital scholarship practitioners to create accessible, stable, long-lasting resources in the humanities.↩

3. Limitations: The full list of types, relations and attribute properties from the RO-Crate specification are not included. Examples shown include `datePublished`, `CreativeWork` and `name`. Contextual entities not related from the RO-Crate (e.g. using inverse relations to a data entity) would not be covered by the single direction `Mentions(R, s)` production rule; see issue #122. The `datePublished(e, date)` rule do not include syntax checks for the ISO 8601 datetime format. Compared with RO-Crate examples, this generated JSON-LD does not use a `@context` as the IRIs are produced unshortened; a post-step could do JSON-LD Flattening with a versioned RO-Crate context.↩