Packaging research data with RO-Crate

This manuscript (permalink) was automatically generated from stain/ro-crate-paper@8a833dd on September 26, 2020.

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-02-2018-823830); EOSC-Life (European Commission H2020-xx-824087)

- **Paul Groth**
  0000-0003-0183-6910 · pgroth
  Informatics Institute, Faculty of Science, University of Amsterdam, NL

Abstract

Structure from ECCB presentation

- Slides: https://doi.org/10.5281/zenodo.4011999

FAIR principles

FAIR principles https://doi.org/10.1038/sdata.2016.18 lay out principles for ensuring research data is Findable, Accessible, Interoperable and Reusable. Key factor the FAIR principles for Interoperability is *machine readability*:

> I1. (meta)data use a **formal**, accessible, shared and broadly applicable **language for knowledge representation**. I2. (meta)data use **vocabularies** that follow FAIR principles I3. (meta)data include qualified **references** to other (meta)data

FAIR not just for data - also for research software, how to make that data and where data came from. https://doi.org/10.3233/DS-190026

Best practice for workflow reproducibility

The use of computational workflows has gained prominence, in particular life sciences, typically combining a chain of open source tools in an analytical pipeline. While the workflows initially may have been used to improve scalability, it can be argued they both assist in making computated data results FAIR, but at the same time raising additional FAIR challenges when considering the workflows as important research artifacts themselves in order to capture and explain the computational method behind an analysis. https://doi.org/10.1162/dint_a_00033

Even when researchers follow current best practice for workflow reproducibility, https://doi.org/10.1016/j.cels.2018.03.014 https://doi.org/10.1016/j.future.2017.01.012 the communication of that outcome 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. Even as researchers the ambition of FAIR reproducible research, it has not yet become common practice.

As a real-life example let's look at a metagenomics article in Nature https://doi.org/10.1038/s41586-019-0965-1, where the authors have gone to extraordinary effort 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 FTP server (60 GB) http://ftp.ebi.ac.uk/pub/databases/metagenomics/umgs_analyses/ and the open source code in GitHub repository https://github.com/Finn-Lab/MGS-gut includes 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 https://twitter.com/soilandreyes/status/1250721245622079488 - rather authors are often penalized for doing so [cite?] as it would work against artificial limits on number of pages and references.

However detailed, for a new researcher who wants to reuse a particular computational method they 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 workflow following the method description). https://doi.org/10.3389/fninf.2017.00069 [Goble R* bruahaha]

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 may would firstly need to install all the software dependencies and reference datasets. This can be a daunting task in itself, which may have to be repeated multiple times as workflows typically are developed at small scale on their desktop computer, scaled up to a local cluster, 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 https://doi.org/10.1007/s41019-017-0050-4 with supporting collaborative efforts like BioConda https://doi.org/10.1038/s41592-018-0046-7, BioContainers [??] and by Linux distributions themselves (Debian Med https://doi.org/10.1186/1471-2105-11-S12-S5) to make more than 7000 software packages available in BioConda alone https://anaconda.org/bioconda/ and 9000 containers in BioContainers https://biocontainers.pro/#/registry. Docker and Conda has 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.

Research Objects - a brief history

The challenge of describing computational workflows was one of the main motivations for the proposal of *Research Objects* https://doi.org/10.1016/j.future.2011.08.004 as first-class citizens for sharing and publishing, by bundling datasets, workflows, scripts, 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 https://doi.org/10.1016/j.future.2011.08.004 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 (the FAIR paper was published 7 years later) that the initial Research Objects approach was strongly targeting Interoperability. [???]

The first implementation of Research Objects in 2009 for sharing workflows in myExperiment https://doi.org/10.1093/nar/gkq429 was based on RDF ontologies http://eprints.soton.ac.uk/id/eprint/267787, building on Dublin Core, FOAF, SIOC, Creative Commons, OAI-ORE and (later) DBPedia to form myExperiment ontologies for describing social networking, attribution and creditation, annotations, aggregation packs, experiments, view statistics, contributions, and workflow components. http://web.archive.org/web/20091115080336/http://rdf.myexperiment.org/ontologies Programmatic access to Research Objects was facilitated with an RDF endpoint that exposed individual myExperiment resources, also queriable from a SPARQL endpoint, both using the myExperiment vocabularies and RDF formats RDF/XML and Turtle.

However, most programmatic users of myExperiment, such as workflow systems and portals, ended up using its separate XML-based REST API; particularly as it provided a

WF4Ever. Preservation.

Research Object ontologies. OAI-ORE, OAC/AO/OAC.

Research Object Bundle - downloadble ROs.

RO-BagIt. archivable ROs

BDBag. scalable ROs.

CWLProv. rerunnable ROs.

Re-use of existing vocabularies. Making lots of new vocabularies. - too many??

Stack becomes very large to learn and use.

RDF as its own worst enemy.

Need for simplicity. Cater for “web developer”.

RO-Crate - Rebirth of Research Object

DataCrate origin

Not workflows!

More traditional data archiving.

“As a researcher…I’m a bit bloody fed up with Data Management” https://cameronneylon.net/blog/as-a-researcher-im-a-bit-bloody-fed-up-with-data-management/

Community building

schema.org as basis and almost sole vocabulary

JSON-LD vs JSON

Tutorial vs Specification

Workflow RO-Crate

.. and BioSchemas profile for Workflows

What is a workflows anyway?

Multiple variants/flavours. Multiple files. Tool vs workflow.

Tooling for RO-Crate

Libraries for developers

Javascript, Python, Ruby.

No need for Java!

Describo - UI for making RO

..

Profiles: Why not SHACL SHeX or JSON-Schema malarchy?

What next??

Bringing back more of the original ideas. Cross-referenced ROs. Nested ROs. Citing ROs. DOIs? Web vs archive.

Other stuff going on - what’s in the stack?

(stuff from Primer on workflow packaging and metadata standards https://docs.google.com/document/d/1XREgfYNi7l4HbdrnXBs7Uv1tMH2AiR435SKjisu4l30/edit#)

Nanopublications

BioCompute Objects (IEEE 2791)

*Prov models

FAIR Digital Objects

Formal definition of RO-Crate

https://en.wikipedia.org/wiki/List_of_logic_symbols
https://en.wikipedia.org/wiki/Mathematical_Alphanumeric_Symbols

https://www.w3.org/TR/2003/NOTE-lbase-20031010/

http://www2003.sztaki.hu/cdrom/papers/refereed/p050/p50-horrocks.html

𝕌 𝕀𝕣𝕚

𝔻 𝕃 𝕊

𝑟𝜎 𝒓𝝈 𝔯𝔬

Propositional Logic
¬ ¬ → ⟶ ↔ ⟷ ∨ ‖ or ∧ & and ⊢ ⊢ proves ⊣ ⊣ ⊕ ≢ xor 𝔻 Domain of predicate ℒ language

Predicate Logic
∀ ∀ ∃ ∃ ∈ ∈ ⊨ ⊨ entails

Modal Logic
□ □ ◊ ◊

Set ⊃ superset / material implication ≡ ⇔

⊤ tautology ⊥ contradiction

∴ therefore ∵ because ∃!

Formalizing RO-Crate in First Order Logic

Below is a brief formalization of RO-Crate as a set of relations in First Order Logic, followed by a mapping to RDF using schema.org and forward-chaining production rules for making JSON-LD.

𝕃 *ro-crate* = { Property(p), Class(c), Literal(x), Describes(R, s) } 𝔻 = 𝕀𝕣𝕚 𝕀𝕣𝕚 ≡ { IRIs as defined in https://tools.ietf.org/html/rfc3987 } ℝ ≡ { real or integer numbers } 𝕊 ≡ { literal strings }

Minimal RO-Crate

RO-Crate(R) ⊨ Root(R) ∧ Describes((R, R) RO-Crate(R) ⊨ hasPart(R, d) ∧ Describes((R, d) ∧ DataEntity(d) RO-Crate(R) ⊨ Describes((R, c) ∧ ContextualEntity(c)

Root(r) → Dataset(r) ∧ published(r, Date) published(e, date) → Literal(date) DataEntity(e) ≡ File(e) ⊕ Dataset(e) Entity(e) ≡ DataEntity(e) ∨ ContextualEntity(e)

Describes(R, s) ⊨ Relation(s, p, e) ⊕ Value(s, p, l) ∀x . Value(o, p, x) → Literal(x) Literal(x) ≡ x ∈ ℝ ⊕ x ∈ 𝕊

Relation(s, p, o) ⊨ Entity(s) ∧ Property(p) ∧ Entity(o) Entity(e) → Metadata(e) Metadata(e) → Class(t) ∧ Describes(R, e)

Mapping to RDF with 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)

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) published(e, date) → Value(e, http://schema.org/datePublished, date)

RO-Crate 1.0 Metadata File Descriptor

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

Forward-chained Production Rules for JSON-LD

Describes(R, S) ∧ Relation(S, P, O) → Describes(R, O) i ∈ 𝕀𝕣𝕚 → i r ∈ ℝ → r s ∈ 𝕊 → "s" Relation(s,p,o) → { "[**???**]": s, p: { "[**???**]": o } } Value(s,p,o) → { "[**???**]": s, p: o } RO-Crate(r) → { "[**???**]": [Describes((r, c)] } R ≡ <./> MetadataFileDescriptor(<ro-crate-metadata.json>) .

Mapping to OAI-ORE

Mapping to OAI-ORE with annotation

References