RO-Index: A survey of Research Object usage

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)

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

Abstract

> This manuscript is **work in progress** and (for now) follows the style of a Study Protocol for F1000Research Registered Reports

For this study we aim to build **RO-Index**, a broad and comprehensive corpus of Research Objects found "in the wild". The proposed methodology follows multiple strands to find the "breeding grounds" of research objects and further describes how Research Objects are selected for inclusion, along with post-processing to build the corpus.

The corpus of Research Objects will primarily be distributed as Open Data, including:

- Identifiers and access URLs
- Extracted manifests and annotation files
- Checksums/references for external data
- Metadata from repository (e.g. Datacite XML)
- Provenance of data gathering and post-processing

Research Objects that cannot be redistributed (e.g. unknown license) will only be examined for aggregates.

A brief set of qualitative and quantitative analytics will then be performed across the overall corpus, in particular to address research questions like:

- Where are Research Objects published?
- Which scientific domains produce Research Objects?
- What serializations are used for making Research Objects?
- What vocabularies are used to describe Research Objects and their content?
- What type of resources to Research Objects contain?
- What kinds of life cycles do Research Objects follow?

Introduction

Protocol

Finding Research Objects

One goal of this work is to determine what kind of artifacts, in practice, can be considered a *research object*. For the purpose of building a corpus we need to have both inclusion and exclusion criteria.

The foundational article on the RO concept is [1] and its workshop predecessor [2]. The Research Object community has maintained lists of initiatives and Research Object profiles which provide curated, although potentially biased, collections of Research Object approaches and implementations.

Declared Research Object usage

In order to determine potential sources of Research Objects we will start with these community lists, but expand based on a literature review by following any academic citation of the before-mentioned Research Object articles to find potential repositories, tools and communities that may conceptually claim to have or make "research objects". This is a broad interpretation that does not expand into general datasets or packaging formats. The list may be expanded by literate search for "Research Object", the RO vocabularies and standard URLs.

Each of the citing articles will then be assessed to see if they have openly accessible research objects that are possible to identify, and ideally retrieve, by building a programmatic crawler. Ideally such access would use an open harvesting protocol like OAI-PMH or ResourceSync, but it is predicted that in the majority of cases custom crawler code will need to be developed per repository, in addition to manual harvesting of identifiers for smaller collections and individual Research Objects.

Keyword searches

7 In addition to this "self-claimed" research object usage we will search in more general repositories by developing a list of keywords like "research object", "robundle" or the RO vocabulary URLs. We will search in at least:

- https://github.com/
- https://gitlab.com/
- http://datacite.org/
- https://zenodo.org/

- https://toolbox.google.com/datasetsearch/
- https://dataverse.harvard.edu/

It is predicted that these searches will yield duplicates, but will be used to find potentially new Research Object sources or free-standing instances.

Archives with manifests

Finally we will consider broadly Open Data repositories of file archives (e.g. ZIP, tar.gz) to inspect for the presence of a *manifest*-like file (e.g. `/manifest.rdf`). For practical reasons this search will be restricted to a smaller selection of public repositories and formats, e.g. Zenodo (20k *.zip Datasets), FigShare ("zip" Datasets), Mendeley Data "zip" File Set.

A list of trigger filename patterns will be developed, including:

- `META-INF/manifest.xml` and `META-INF/container.xml` from EPUB Open Container Format
- `manifest.xml` from COMBINE archives [3]
- `.ro/manifest.rdf` from RO Hub [4]
- `.ro/manifest.json` from Research Object Bundle [5]
- `metadata/manifest.json` from RO-Bagit and BDBag [6;]
- `CATALOG.json` from DataCrate [7]
- `ro-crate-metadata.jsonld` from RO-Crate [8]

It is predicted that most of the archive files will *not* contain such a manifest, therefore they can be inspected "on the fly" by the crawler without intermediate storage, to first detect a short-list of archives that contain a manifest-like file. These can then be downloaded in full for further inspection. File-name matching will inspect potential sub-directories, e.g. to detect `nested/data/manifest.xml`, but will classify these archives differently from direct matches.

Candidate sources

For each candidate source we will collect and assess:

- Date assessed
- Assessed by
- URL
- Name
- Estimate # ROs
- Estimate # users
- Maintainer/publisher
- Community links (if any)
- RO profile/format (if any)
- Identifier scheme(s) (if any)
- Persistence/Versioning (if any)

Then for each candidate source we will evaluate:

- Accessability - can we retrieve RO and/or their metadata
- License - permissions and/or restrictions to redistribute the ROs and/or their metadata
- Feasibility - can we programmatically retrieve all ROs (or just a sample)?
- Duplication - could the "same" RO be present by multiple identifiers or in other repositories?
- Self-identified - are Research Objects classified as such (or using similar terminology)?

We may contact the provider or maintainer to expand on these questions if unclear from public information, however we are not conducting a formal survey, as our main interest lays in the machine-readable information from the research objects themselves.

We will finally form a shortlist of sources for further harvesting, considering:

- Programmatically access (or interesting enough to warrant manual access)
- Diversity - might this source be different from the majority of sources?
- Legality - are we allowed to retrieve ROs (or their identifiers and metadata?)
- Confidentiality - are the research objects accessible to the public? (anonymous access or access by 'fresh' user registration)

Handling personally identifiable information

Research Objects may, by their nature, contain information about people and their research activities. It is therefore important that our data collection, processing and potential re-distribution is in consistent with the General Data Protection Regulation (GDPR). To this end we will evaluate:

- Does the source have a GDPR-compliant privacy policy or equivalent?
- Is personally identifiable information contained by identifier (e.g. username)?
- May personally identifiable information be contained by the Research Object manifest/description
- May personally identifiable information be contained by the Research Object files/content?
- Does the RO (or the metadata) have a license that permits redistribution and attribution, e.g. Creative Commons Attribution 4.0 (CC-BY)?

Evaluating this may require retrieving research objects in the first place, but particular care will be taken to classify Research Objects and their sources according to the above evaluation in order to filter information that can progress to be part of the Open Data RO-Index corpus. This forms a staged inclusion list:

1. Unfiltered list of identifiers for a source will be shared if the identifiers tend not to include personally identifiable information
2. Metadata will be shared if it is accessible and does not tend to include personally identifiable information

3. Metadata and identifier will be shared if an open attribution-permitting license is indicated (or implied by site)
4. Content/files will be shared if accessible and an open license is indicated (or, for archives, implied by archive license)

Note: In the above, "tend to" will be determined manually by inspecting a smaller subset of typically 10 research objects. The selection will aim to approximate a simple random subset, but may need to be expanded to take into account the overall diversity of ROs at the source, e.g. date, authors, subsystem, formats. The identifiers of the ROs of this subset will be recorded, along with a description of how the subset was selected.

The inclusion list may be further restricted based on findings from further processing (e.g. a repository is found to distribute sensitive data).

It is worth noting that compliance with open licenses like Creative Commons Attribution 4.0 (CC-BY) or Apache License 2.0 **require** attribution to be propagated (if present). Attribution may sometimes take the form of a URL, identifier, project or organization which do not directly identify a person.

The inclusion list will form different subsets of Research Objects:

1. Identified Research Objects
2. "Non-sensitive" (but potentially closed) metadata
3. Open metadata (potentially personally identifiable)
4. Open content (potentially personally identifiable)

Data for any excluded Research Objects will only be kept for the purpose and duration of this study on computer infrastructure managed by The University of Manchester. Data from excluded Research Objects will only be used for non-person-identifiable aggregated results (e.g. number of CSV files) and broad categorization (e.g. vocabularies used in metadata).

The identifiers from category 1, metadata from category 3 and data from category 4 will be shared in the public Open Data repository Zenodo according to Zenodo's policies. Metadata from category 3 and 4 above may be exposed for programmatic querying (e.g. SPARQL) or converted to other formats. No additional linking with internal and external data sources will be performed, although the collected Research Objects may already contain such links (e.g. https://orcid.org/ identifiers of authors); an exception to this rule is that linking will be permitted to detect duplicate Research Objects across multiple sources, and to access resources clearly *aggregated* as part of the Research Object.

For GDPR purposes the *Data Controller* is The University of Manchester, data subjects may contact `info@esciencelab.org.uk` for any enquiries, such as to request access to data about themselves, or to request update or removal of personally identifiable information.

Pre-identified data sources

Proto-research objects

- myExperiment packs
- COMBINE archives [3,9]
- VoID datasets http://www.openphacts.org/specs/2013/WD-datadesc-20130912/ [10,3]
- DataONE Data packages [11]
- DataLad https://www.datalad.org/datasets.html [12]
- BloodHound / Global Biodiversity Information Facility GBIF Darwin Core Archives [13] e.g. [14] https://www.gbif.org/dataset/search
- Crystallographic Binary File CBS/imgCIF [15] - instrument-specific metadata. HBF

ORE-based research objects

- CWL Viewer https://view.commonwl.org/workflows [16]
- RO Bundle https://w3id.org/bundle/2014-11-05/ [5]
- Workflow PROV corpus [17]
- CWLProv 10.1093/gigascience/giz095 aka [18]
- http://www.rohub.org/ [4]
- http://rohub.linkeddata.es/
- SEEK: https://fairdomhub.org/investigations
- BDBags with MinID [6;]
- Zenodo e.g. [19]
- Mendeley Data eg [20]
- Maven https://repository.mygrid.org.uk/artifactory/ops/org/openphacts/data/
- DocumentObject https://github.com/binfalse/DocumentObjectCompiler/
- GitHub search
- EOSC-Life (too early?)

Software/container-based research objects

- https://sci-f.github.io/ [21]
- https://frictionlessdata.io/specs/data-package/
- The Journal of Research Objects http://jro.world/ (see also presentation
- Tonkaz https://docs.google.com/presentation/d/1YRKCM1KwHyNOz7B6AP2j5qIHjl7Ew9CnAtvXA1xSeNY/edit#slide=id.g419a3a9d09_1_93
- ActivePaper / HDF5 [22] > HDF5 dataset attributes are used to store metadata, including a dataflow graph that records provenance (requirement 5), but also creation time stamps and a data type indicator distinguishing references and executable code from "plain" datasets.

2nd generation ROs

- DataCrate: https://github.com/UTS-eResearch/datacrate/blob/master/spec/1.0/data_crate_specification_v1.0.md#examples
- RO-Crate: https://data.research.uts.edu.au/examples/ro-crate/0.2/

Manifest formats

A key characteristic of a Research Object is the presence of a *manifest* that describes and relates the content. However, multiple potential formats and conventions have emerged for how to serialize such a format. (..)

Proposed data gathering workflow

The overall data gathering workflow is envisioned as:

1. Traverse repository (or one of its sub-sections) using API like OAI-PMH
2. Filter for entries that have an archive-like file type (e.g. ZIP, tar.gz)
3. Retrieve entry's Datacite-like metadata from repository (e.g. DOI, author, license)
4. Start downloading archive
5. Stream archive though a utility like sunzip to list filenames within
6. Record filenames mapped to identifier
7. Select entries which have a manifest-like file in list
8. Re-download selected archives
9. Extract manifest(s) from archives
10. Classify manifests based on format and vocabulary (e.g. RDF/XML using ORE-OAI)
11. Record provenance of data gathering

Post-processing workflow:

1. Convert manifests to a unified RDF format (e.g. N-Triples)
2. Populate quad store (e.g. Apache Jena) with converted manifests
3.

https://zenodo.org/communities/ro/?page=1&size=20

https://developers.zenodo.org/#metadata-formats

Prototype workflow

A prototype workflow is being developed using Common Workflow Language [23], figure 1 shows how a a community sub-section of the Zenodo repository is being inspected to list the filenames contained within its downloadable ZIP files.

Figure 1: **CWL workflow: List ZIP content for Zenodo community** Visualization by CWL Viewer
https://w3id.org/cwl/view/git/4360a062e7cff5aadacbf401e8e743a660657680/code/data-gathering/workflows/zenodo-zip-content.cwl

In brief the prototype workflow consists of these steps:

1. For a given Zenodo community, e.g. https://zenodo.org/communities/ro, retrieve its OAI-PMH entries using DataCite v4 XML
   e.g. https://zenodo.org/oai2d?verb=ListRecords&set=user-ro&metadataPrefix=datacite4
2. Extract the URI for the Zenodo record, e.g. <oai:zenodo.org:1484341>
3. Search-replace to build the URI of the the corresponding Zenodo landing page, e.g. https://zenodo.org/record/2838898
4. Retrieve the HTML of the landing page
5. Extract the link relations [24] of type alternate to find the download links
6. Filter for download links that end with `*.zip`
7. Retrieve ZIP file
8. List filenames within ZIP file
9. Detect filenames like `manifest.rdf` `eml.xml` or `meta.xml` within list and return original Zenodo URI or `null` *(in development)*

The workflow and its components have been tested with the reference implementation `cwltool` [25] which can provide rich provenance captured in CWLProv research objects [26]

In developing this prototype several challenges where immediately detected:

- The OAI-PMH records contain substansive DataCite information, but do not include the links to the Zenodo record or the ZIP downloads. The identifiers for records is in a form that is specific to OAI-PMH, e.g. `oai:zenodo.org:1310621` and had to be rewritten to a URI using the same record number.
  - The Zenodo API does include these additional URIs - but is not available for anonymous use as it requires a registration token. Distributing this token as part of the workflow provenance would give access to the author's Zenodo account.
- The HTML landing page provides links to structured metadata, e.g. as JSON-LD using Schema.org https://zenodo.org/record/1310621/export/schemaorg_jsonld but the structured data is included in a `<pre>` block within HTML (for copy-pasting) and not programmatically accessible. HTTP content-negotiation does not give direct access to the structured data. There are no `alternate` links from the HTML landing page to these "export" pages, so their URIs would have to be manually constructed.
  - The JSON-LD does include links to downloads of ZIP files etc, but only if the Record is of type *Dataset* in Zenodo, which in JSON-LD is represented as a DataDownload with a contentUrl
- To list filename contained in a ZIP file, the whole file must be downloaded because in the ZIP format the table of content is written at the end of the file
  - The host providing Zenodo downloads do not support the HTTP Range request header [27], so it is not possible to download say only the last 128 kB of the ZIP file.
  - Some of the dataset downloads are larger than 1 GB
  - `cwltool` saves all intermediate values until the end of the workflow
- To preserve disk space use during download of ZIP file only, an initial idea was explored of using the CWL streamable feature so that the output of `curl` could be passed to the utillity sunzip which can extract/inspect ZIP files in a streaming fashion
  - It was found that while the sunzip tool supports streamable extraction and testing based on the intermediate file entry records of the ZIP file, it does not decide or list the filenames before the end of the download. Thus the output of `sunzip -t` (test extract) itself is not streamable
  - An experimental new feature `sunzip -l` was attempted, where file names of records are printed as they are encountered. This is at the risk of printing files that have subsequently been deleted from the ZIP file and do not appear in the table of content. This was considered a small risk as the primary purpose was to detect ZIP files containing "manifest-like" files and it was assumed that most deposited ZIP files had been created in a one-off operation and would not have inconsistent file records.
  - The experimental feature however detected that several files found in Zenodo are written with a "deferred length" - the intermitting ZIP records do not record their length and so `sunzip` is unable to find the offset to the next record without actually decompressing the stream.
  - It was found that `cwltool --parallel` did not seem to start the subsequent step and thus did not facilitate the `streamable` feature. The ZIP files were still saved to disk, and all the ZIP file downloads where completed before any of the ZIP extraction was started. Future work will explore using the implementation `toil` which have been argued to have better support for concurrency.

Estimating bandwidth and storage requirements

In order to estimate the bandwidth and storage requirements for executing the above prototype workflow across the whole of Zenodo, a shell script approach was used to retrieve and analyse the JSON for each Zenodo record, which include information on downloadable filename, extension and file size.

To retrieve these it was deemed necessary to use the undocumented parts of Zenodo API. From the Zenodo source code it was identified that the REST template `https://zenodo.org/api/records/{pid_value}` could be used with `pid_value` as the numeric part from the OAI-PMH identifier, e.g. for `oai:zenodo.org:1310621` the Zenodo JSON can be retrieved at https://zenodo.org/api/records/1310621.

The JSON API supports content negotiation, the content-types supported as of 2019-09-20 include:

- `application/vnd.zenodo.v1+json` giving the Zenodo record in Zenodo's internal JSON schema (v1)
- `application/ld+json` giving JSON-LD Linked Data using the http://schema.org/ vocabulary
- `application/x-datacite-v41+xml` giving DataCite v4 XML [28]
- `application/marcxml+xml` giving MARC 21 XML

Using these (currently) undocumented parts of the Zenodo API thus avoids the need for HTML scraping while also giving individual complete records that are suitable to redistribute as records in a filtered dataset.

This preliminary exploration will be adapted into the reproducible CWL workflow. Below is a bash transcript. Execution time was about 3 days from a server at the University of Manchester network on a single 1 GBps network link. The script does:

- Retrieve each of the first 3.5 million Zenodo records
  as Zenodo JSON by iterating over possible numeric IDs (current maximum ID `3450000` estimated from "Recent uploads")
- Filter list to exclude records that are not found, moved or deleted. The presence of the key `conceptrecid` is used as marker.
- Use jq to ensure the JSON is on a single line
- Join the JSON files using the ASCII Record Separator (RS, `0x1e`) to make a `application/json-seq` JSON text sequence [29] stream
- Save the JSON stream as a single compressed file using `xz`
- From the JSON, select those records that indicate `{ "metadata": { "access_right": "open"} }`
- Select `{ "files": {"type": "zip"} }`
- Summarize `{ "files": {"size": 12345} }` (bytes)

```
## Naively Download first 3450000 records (many of are 404)
# TODO: Run from OAI-PMH ids to avoid 404
# TODO: Respect Retry and X-Ratelimit headers (this only overrun once)
# TODO: Parallelize 2-3 threads so it does not take 3 days
curl --retry 10 -H "Accept: application/vnd.zenodo.v1+json" \
  'https://zenodo.org/api/records/[1-3450000]' -o "record_#1.json"

# Ignore "404" etc.
# Convert to
find . -maxdepth 1 -type f -name 'record*json' | \
  xargs -n32 cat | \
  jq -c .  | \
  grep conceptrecid | \
  sed `echo -e "s/^/\x1e/g"` | \
  xz -4 -T5 > records-with-conceptrecid.jsonseq.xz

xzcat records-wtih-conceptrecid.jsonseq.xz |  \
  jq '. | select(.metadata.access_right == "open") | .files[]? |
      select(.type == "zip") | reduce .size as $s (0; . + $s)' | \
  jq --slurp add
# 27620751244752
```

This indicates that downloading all the *.zip files is about 27 TiB.

```
jq -n '27620751244752 / 1024 / 1024 / 73.9 / 3600 / 24'
# 4.125507608812889
```

Downloading all will take at least 4 days assuming 73.9 MBit/s as measured using `wget` of a [9 GB file](#) - it is predicted that actual download time may be doubled because of the effect of latency on shorter downloads, which will not be able to saturate the link speed before the download is complete. For comparison downloading the JSON files, each about 1 kB, took 3 days, so a realistic estmiate is 7 days to download.

It was found that only a small subset of downloads are over 30 MB. Keeping all the ZIP files <30MB will require about 300 GB.

The machine used for this experiment has about 1 TB free across 2x 1.8TB disks.

Therefore the suggestion is to split the download list into two subsets, a) with many small ZIP files which are kept b) large ZIP files which are processed by streami

(random notes below)

```
TODO:

- Create PROV entities for each record, download, manifest, datacite etc.
https://doi.org/10.1016/j.websem.2015.04.001

- Can we get data/stats for all of Zenodo? Start download to fill .cache


Some of the commands..

# TODO: Consistency check of records
find . -maxdepth 1 -type f -name 'record*json' | xargs grep ^.."status.*404" | cut -d ":" -f 1 | xargs mv -t 404/


### TODO: Redo as Jupyter notebook

### Let's check the size of the *.zip files

find . -maxdepth 1 -type f -name 'record*json' | xargs -n3 cat | jq '.files[]? | select(.type == "zip") | reduce
.size as $s (0; . + $s)  ' > zipsizes.txt


## Some quick estimates on workload and network/disk requirements :

# How many terabytes to download?
cat zipsizes.txt | jq --slurp 'reduce .[] as $s (0; . + $s) | ./1024 /1024 /1024 / 1024'
3.8408555243122464
# (at 1008098/3450000 downloaded)

# So in total we will need to download about 13 TB
stain@ondex2:/tmp/zenodo$ cat zipsizes.txt | jq --slurp 'reduce .[] as $s (0; . + $s) | ./1024 /1024 /1024 / 1024
/1008098*3450000'
13.144507338450477

# Downloading 19 MB takes 0.8 s
# stain@ondex2:/tmp$ time wget https://zenodo.org/api/files/36ee07d5-7c07-4382-8463-4502d98148a4/s10.zip
--2019-09-15 00:13:19--  https://zenodo.org/api/files/36ee07d5-7c07-4382-8463-4502d98148a4/s10.zip
Resolving zenodo.org (zenodo.org)... 188.184.65.20
Connecting to zenodo.org (zenodo.org)|188.184.65.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20233334 (19M) [application/octet-stream]
Saving to: 's10.zip.1'

s10.zip.1                                          100%
[=====================================================================================================>]
  19.30M  59.6MB/s    in 0.3s

2019-09-15 00:13:20 (59.6 MB/s) - 's10.zip.1' saved [20233334/20233334]


real    0m0.823s
user    0m0.056s
sys     0m0.128s

## Only 6000/31370 files or so are more than 30 MB.
stain@ondex2:/tmp/zenodo$ cat zipsizes.txt | jq 'select(. > 30*1024*1024)' | wc -l
6318

## Only 655/31370 are more than 1 GB
stain@ondex2:/tmp/zenodo$ cat zipsizes.txt | jq 'select(. > 1024*1024*1024)' | wc -l
655

## Downloading a file of 10 GB runs at 73.9 MB/s avg
(tested at Sunday 2019-09-15 01:24 - so over-ideal timing)

stain@ondex2:/tmp$ time wget https://zenodo.org/api/files/2cdaea21-80be-48b9-9ab2-259671795f7a/br20832_cores.zip
--2017-09-15 00:22:52--  https://zenodo.org/api/files/2cdaea21-80be-48b9-9ab2-259671795f7a/br20832_cores.zip
Resolving zenodo.org (zenodo.org)... 188.184.65.20
Connecting to zenodo.org (zenodo.org)|188.184.65.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10081056983 (9.4G) [application/octet-stream]
Saving to: 'br20832_cores.zip'

br20832_cores.zip                                  100%
```

```
[================================================================================>]
   9.39G    101MB/s    in 2m 10s

2019-09-15 00:25:03 (73.9 MB/s) - 'br20832_cores.zip' saved [10081056983/10081056983]


real    2m10.634s
user    0m31.100s
sys     0m55.288s

## This machine is on a single Gbit link (capable of two links) with MTU 1500
# in server room of Computer Science UNIMAN - jumboframes not supported by network




## Absolute fastest we can download 13 TB?
stain@ondex2:/tmp/zenodo$ jq -n '13.144507338450477 * 1024 *1024 / 73.9 / 3600 / 24'
2.158668954374506
# aka 2 days

# how long would Non-parallel download of the small files take?

cat zipsizes.txt | jq 'select(. <= 30*1024*1024)' | wc -l
25052

(tip, 1008098*3450000 is the scaling factor as these estimates are done after
downloading about 30 % of zenodo records -- WARNING: Those are not representative but
are the 30% oldest. )

stain@ondex2:/tmp/zenodo$ jq -n '25052 * 0.823 / 1008098*3450000 / 3600 / 24'
0.816

# Should take less than a day? (That would means API download is slower.. )

So in total 3 days for all zip downloads. No need for special measures.

# All the small ZIP files fit on single disk
stain@ondex2:/tmp/zenodo$ cat zipsizes.txt | jq -s '.[] | select(. <= 30*1024*1024)' | jq -s 'reduce .[] as $s
(0; . + $s)  / 0.30/ 1024/1024/1024'
322.71499813844764

# .. but just about!
stain@ondex2:/tmp/zenodo$ df -h .
Filesystem                  Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup00-tmp  296G   46G  235G  17% /tmp

# (total disk space is 1.8T*2, about 1 TB free)

## TODO: Decide on threshold for data that should be kept on disk

## TODO: Can we use Isolon for local storage..? No bandwidth benefit
# but perhaps latency improvement. Might have just 300 GB quota left

# //nasr.man.ac.uk/epsrss$/snapped/replicated/goble/methodbox
#                        600T  555T   46T  93% /rds/methodbox


# Use methodbox.cs.man.ac.uk ...? More diskspace (0.8 TB * 2)
# the 1TB of Isilon storage has now been setup and is available for use.

## artifactory Used: 790.6 GB
## methodbox 189G
## uh.. 1 TB alrady


# Concatinate *.json into a single file for future use with jq
# Note that this file is not JSON, but application/json-seq
# https://tools.ietf.org/html/rfc7464
# -- for use wih jq use jq --seq
find . -maxdepth 1 -type f -name 'record*json' | xargs -n32 cat | jq -c .  | sed `echo -e "s/^/\x1e/g"` | xz >
records.jsonseq.xz

# Find filenames of uploads
xzcat < records.jsonseq.xz | jq --seq --raw-output '.files[]?.key'  > deposit_filenames.txt
```

```
# Build a jsonseq with only json records containing "conceptrecid" (avoiding "moved" and "not found" etc)
# (avoids the previous naive 404 filtering)


# Look for *.zip file that are open to download

xzcat records-wtih-conceptrecid.jsonseq.xz |  jq '. | select(.metadata.access_right == "open") | .files[]? |
select(.type == "zip") | reduce .size as $s (0; . + $s)' | jq --slurp add

27620751244752

(aka 27 TiB)
```

Conclusions/Discussion

Data (and Software) Availability

Author contributions

- Conceptualization:
- Data Curation:
- Formal Analysis:
- Funding Acquisition: SSR, CAG
- Investigation:
- Methodology:
- Project Administration:
- Resources: CAG, SSR
- Software: SSR
- Supervision: PG
- Validation:
- Visualization:
- Writing – Original Draft Preparation: SSR
- Writing – Review & Editing:

Competing interests

Grant Information

References

1. **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

2. **Why Linked Data is Not Enough for Scientists**
Sean Bechhofer, John Ainsworth, Jiten Bhagat, Iain Buchan, Philip Couch, Don Cruickshank, David De Roure, Mark Delderfield, Ian Dunlop, Matthew Gamble, … Shoaib Sufi
2010 IEEE Sixth International Conference on e-Science (2010-12) https://doi.org/cv5tzk
DOI: 10.1109/escience.2010.21

3. **COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project**
Frank T Bergmann, Richard Adams, Stuart Moodie, Jonathan Cooper, Mihai Glont, Martin Golebiewski, Michael Hucka, Camille Laibe, Andrew K Miller, David P Nickerson, … Nicolas Le Novère
BMC Bioinformatics (2014-12) https://doi.org/gb8wc5
DOI: 10.1186/s12859-014-0369-z · PMID: 25494900 · PMCID: PMC4272562

4. **ROHub — A Digital Library of Research Objects Supporting Scientists Towards Reproducible Science**
Raúl Palma, Piotr Hołubowicz, Oscar Corcho, José Manuel Gómez-Pérez, Cezary Mazurek
Communications in Computer and Information Science (2014) https://doi.org/gf5m6p
DOI: 10.1007/978-3-319-12024-9_9

5. **Research Object Bundle 1.0**
Stian Soiland-Reyes, Matthew Gamble, Robert Haines
Zenodo (2014-11-05) https://doi.org/gf5m6k
DOI: 10.5281/zenodo.12586

6. **Reproducible big data science: A case study in continuous FAIRness**
Ravi Madduri, Kyle Chard, Mike D'Arcy, Segun C. Jung, Alexis Rodriguez, Dinanath Sulakhe, Eric Deutsch, Cory Funk, Ben Heavner, Matthew Richards, …
Ian Foster
PLOS ONE (2019-04-11) https://doi.org/gf5m6s
DOI: 10.1371/journal.pone.0213013 · PMID: 30973881 · PMCID: PMC6459504

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

8. **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

9. **Ro-Combine-Archive**
Stian Soiland-Reyes, Matthew Gamble
Zenodo (2014-04-28) https://doi.org/gf5m6t
DOI: 10.5281/zenodo.10439

10. **Applying linked data approaches to pharmacology: Architectural decisions and implementation**
Gray Alasdair J.G., Groth Paul, Loizou Antonis, Askjaer Sune, Brenninkmeijer Christian, Burger Kees, Chichester Christine, Evelo Chris T., Goble Carole, Harland Lee, … Williams Antony J.
Semantic Web (2014) https://doi.org/gf5m6j
DOI: 10.3233/sw-2012-0088

11. **Preserving Reproducibility: Provenance and Executable Containers in DataONE Data Packages**
Bryce Mecum, Matthew B. Jones, Dave Vieglais, Craig Willis
2018 IEEE 14th International Conference on e-Science (e-Science) (2018-10) https://doi.org/gf5m6q
DOI: 10.1109/escience.2018.00019

12. **DataLab**
Yang Zhang, Fangzhou Xu, Erwin Frise, Siqi Wu, Bin Yu, Wei Xu
Proceedings of the 2nd International Workshop on BIG Data Software Engineering - BIGDSE '16 (2016) https://doi.org/gf7hv6
DOI: 10.1145/2896825.2896830

13. **The GBIF Integrated Publishing Toolkit: Facilitating the Efficient Publishing of Biodiversity Data on the Internet**
Tim Robertson, Markus Döring, Robert Guralnick, David Bloom, John Wieczorek, Kyle Braak, Javier Otegui, Laura Russell, Peter Desmet
PLoS ONE (2014-08-06) https://doi.org/f6n8jm
DOI: 10.1371/journal.pone.0102623 · PMID: 25099149 · PMCID: PMC4123864

14. **Natural history specimens collected and/or identified and deposited.**
Claudia Baider
Zenodo (2019-09-12) https://doi.org/gf75d8
DOI: 10.5281/zenodo.3405730

15. **Specification of the Crystallographic Binary File (CBF/imgCIF)**
H. J. Bernstein, A. P. Hammersley
International Tables for Crystallography (2006-10-01) https://doi.org/b27mm3
DOI: 10.1107/97809553602060000729

16. **CWL Viewer: the common workflow language viewer**
Mark Robinson, Stian Soiland-Reyes, Michael R. Crusoe, Carole Goble
F1000Research (2017) https://doi.org/cbq2
DOI: 10.7490/f1000research.1114375.1

17. **A workflow PROV-corpus based on taverna and wings**
Khalid Belhajjame, Jun Zhao, Daniel Garijo, Aleix Garrido, Stian Soiland-Reyes, Pinar Alper, Oscar Corcho
Proceedings of the Joint EDBT/ICDT 2013 Workshops on - EDBT '13 (2013) https://doi.org/gf5m6r
DOI: 10.1145/2457317.2457376

18. **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
Zenodo (2019-07-15) https://doi.org/gf5tg8
DOI: 10.5281/zenodo.1208477

19. **W2Share Case Study: Workflow Research Object (Wro)**
Lucas Carvalho, Claudia Bauzer Medeiros
Zenodo (2018-10-18) https://doi.org/gf5m6m
DOI: 10.5281/zenodo.1465897

20. **CWL run of Alignment Workflow (CWLProv 0.6.0 Research Object)**
Stian Soiland-Reyes
Mendeley (2018-12-04) https://doi.org/gf5m6h
DOI: 10.17632/6wtpgr3kbj.1

21. **The Scientific Filesystem**
Vanessa Sochat
GigaScience (2018-03-13) https://doi.org/gdwq7f
DOI: 10.1093/gigascience/giy023 · PMID: 29718213 · PMCID: PMC5952957

22. **ActivePapers: a platform for publishing and archiving computer-aided research**
Konrad Hinsen
F1000Research (2015-07-14) https://doi.org/gfrkvv
DOI: 10.12688/f1000research.5773.3 · PMID: 26064469 · PMCID: PMC4448745

23. **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

24. **Web Linking**
M. Nottingham
RFC Editor (2017-10) https://doi.org/gf8jcd
DOI: 10.17487/rfc8288

25. **common-workflow-language/cwltool: cwltool 1.0.20190815141648**
Peter Amstutz, Michael R. Crusoe, Farah Zaib Khan, Stian Soiland-Reyes, Manvendra Singh, Kapil Kumar, Anton Khodak, John Chilton, Thomas Hickman, Boysha, ... Ryan Spangler
Zenodo (2019-08-15) https://doi.org/gf7s2s
DOI: 10.5281/zenodo.3369238

26. **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
Zenodo (2019-05-23) https://doi.org/gf7s2r
DOI: 10.5281/zenodo.3196309

27. **Hypertext Transfer Protocol (HTTP/1.1): Range Requests**
R. Fielding, Y. Lafon, J. Reschke (editors)
RFC Editor (2014-06) https://doi.org/gf8jcb
DOI: 10.17487/rfc7233

28. **DataCite Metadata Schema Documentation for the Publication and Citation of Research Data v4.0**
DataCite Metadata Working Group
DataCite e.V. (2016) https://doi.org/gf8jcf
DOI: 10.5438/0012

29. **JavaScript Object Notation (JSON) Text Sequences**
N. Williams
RFC Editor (2015-02) https://doi.org/gf8jcc
DOI: 10.17487/rfc7464