PROTOCOL – Availability of Open Access Metadata from Open Journals – A case study in DOAJ and Crossref V.5

Version 5

Aug 09, 2022

Davide Brembilla[1], Chiara Catizone[1], Giulia Venditti[1]

[1]Alma Mater Studiorum - Università di Bologna

1 *Works for me* Share

dx.doi.org/10.17504/protocols.io.kxygxz7ywv8j/v5

Open Science 2021/2022

Davide Brembilla

ABSTRACT

This protocol is for the research about the availability of Metadata from Open Journals in the DOAJ in Crossref.
The goal is to find out how many papers from DOAJ journals are available on Crossref, whether their metadata is available and the origin of their references' DOIs. This will provide us a clearer picture about the state of Open Access research.

Our research involves the articles from Open Access journals in DOAJ and their data on Crossref. We scouted the availability of these articles' reference lists, the presence of IDs such as DOIs and the entities responsible for their specification. This analysis was carried using DOAJ article metadata dump, then modified and enriched thought Crossref's APIs DOI requests. The data will be further analysed to verify the distribution of the provenance of the articles' DOIs.

DOI

dx.doi.org/10.17504/protocols.io.kxygxz7ywv8j/v5

PROTOCOL CITATION

Davide Brembilla, Chiara Catizone, Giulia Venditti 2022. PROTOCOL – Availability of Open Access Metadata from Open Journals – A case study in DOAJ and Crossref. **protocols.io**
https://protocols.io/view/protocol-availability-of-open-access-metadata-from-ced6ta9e
Version created by Davide Brembilla

KEYWORDS

Open Science

CREATED

Jul 27, 2022

LAST MODIFIED

Aug 09, 2022

OWNERSHIP HISTORY

Aug 09, 2022 Davide Brembilla

PROTOCOL INTEGER ID

67742

GUIDELINES

This methodology cleans and populates the batches of articles downloadable from the DOAJ website and populates them with the infromations from Crossref's API.
All the software to perform this methodologies can be found in the Github Repository for it. All the software is distributed under the MIT licence.
To describe the data in input and output, we produced a Data Management Plan.
The main software is in **multithread_populating.py**, while **main.py** is the script used to launch the query from the command line. To populate, we also used the **batches_cleaner.py** and **journals_cleaner.py** softwares. At the end, we used also the **populator.py** script to add the missing details and the **data_viz.ipynb** notebook to create visualisations.
If you want to query Crossref through a DOI, you can create a **populateJson** instance and query the API with just the DOI. If you are interested in more complete access to the API, you can find here a more complete list of libraries that have a wider scope; another useful software that was of inspiration for the one used in here is oc_graphenricher.

MATERIALS TEXT

All the software to perform this methodologies can be found in the Github Repository for it. All the software is distributed under the MIT licence.
This methodology cleans and populates the batches of articles downloadable from the DOAJ website and populates them with the infromations from Crossref's API.

SAFETY WARNINGS

Querying Crossref may be a long process, varying depending on your connection and on the power of your PC and internet connection.

BEFORE STARTING

All the software to perform this methodologies can be found in the Github Repository for it. In order to use it, you will need Python 3 as well as install the libraries used in the **requirements.txt** file. To install them, you can use the command:

```
pip install -r requirements.txt
pip3 install -r requirements.txt
```

‰ Data Gathering

1 Download DOAJ articles metadata

For starting our research, we first focused on retrieving and cleaning DOAJ articles' data as we identified it as the starting point from which we could look for an answer to our research questions.

1.1 We downloaded the DOAJ public data dump containing **article metadata** in tar.gz format and extracted the files.

The dump is structured as a single directory of the form **doaj_article_data_[date generated]** where 75 files are listed with names of the form **article_batch_[number].json**. Each file contains up to 100,000 records for a total size of 270Mb.

1.2 To decrease the size of each batch, we filtered each key to get only the information useful for our research: 5m

- DOI of the article
- year of publication
- ISSNs journal they belong to

To do so we used the **batches_cleaner.py** software.

```
#Scripts to execute:
py -m batches_cleaner doaj_article_data_[date generated]
#windows
python3 -m batches_cleaner doaj_article_data_[date
generated] #macos
```

Result example

```
{
"doi": {
"year": "XXXX",
"issns": [
    "XXXX-XXXX",
    "XXXX-XXXX"
        ]
    },
...
```

2 Collecting Crossref data

1w

To better reply to our research questions, the cleaned dump had to be enriched with information retrieved by Crossref REST API. We used the **main.py** script that launches the **multithread_populating.py** software, both developed in python.

All methods developed for managing Crossref requests are listed as follows:

2.1 To run the whole software, we launched from the shell the following command:

```
py -m main [path to files] #windows
python3 -m main [path to files] #macos
```

The result of populating will be saved in /temp/completed

2.2 **populate method.**

Populate iterates over all files in the directory **doaj_article_data_[date generated]** controlling if they are temporary files. In the end, it produces a JSON file as output named **batch.json** in the **output** directory.

2.3 **_read_json method** iterates over articles stored in **article_batch_[number].json.**
ISSNs are here used as indexing numbers for the DOIs, grouping them under

the journal they belong to.

If the response status code is not equal to 200 the DOI is not present in Crossref, so we add those key values:
- "'crossref': 0
- "reference": 0
- "type": none

In other cases, we read the response message and we add those key values:
- "crossref": 1
- "reference": 0
- "type": resource's type

Then, if the response presents reference values we read the reference list to read reference article information. Id DOI is present we add this key value to our record:
- "doi": "article-doi"
- "doi": "not-specified" (if not present)

Then we look for the entities responsible for DOI specification that crossref saved under the key "doi-asserted-by". Consequentially, we add to reference informations this key value:
- "doi-asserted-by": "publisher"
- "doi-asserted-by": "crossref"
- "doi-asserted-by": "not-specified" (if not present)

If the response is not equal to 200 or 404, the algorithm stops and stores the wrong value to search recursively.

2.4 For each article DOI, we sent a request through the **query_crossref method**. As the process could encounter some errors we coded our software in a way that saves a temporary file in the directory /temp. This latter shrewdness avoids us losing any progress we made, as the whole project was already long to undertake. During this process, we normalized the doi string through the **normalised function** that tourns dois in the format: 10."prefix"/"suffix".

query_crossref method launches requests to Crossref REST API in the format:

https://api.crossref.org/works/{doi}

The method uses *requests_cache*, *backoff*, and *ReadTimeout* to avoid getting blocked and speed up the process.

```
def__init__(self) -> None:
    requests_cache.install_cache('multithread_cache')
    self.api = "https://api.crossref.org/works/"

    @backoff.on_exception(backoff.expo,
requests.exceptions.ReadTimeout,
max_tries=20)
def query_crossref(self, doi):
        if normalise(doi) == 'False':
            return 1234, doi
        query = self.api + normalise(doi)

        #time.sleep(random.randint(1,3))
        try:
            req = requests.get(query, timeout=60)
        except:
            return 1234, doi
        return req, doi
```

3 To compute files compatible with data analysis libraries, we used the **stats** script, which creates CSV tables that will be used to create the final pickle file. each row of the table will contain the information of one article.
To launch it on the terminal the command is :

```
py -m stats path #windows
python3 -m stats path #macos
```

4 As we recognised some valuable information was missing from DOAJ's cleaned articles, a third researcher added the additional information coming from DOAJs' journals data and double-checked the work already done on the cleaned article's files.

Then, we group data for PISSN and EISSN and we add:

- Country of provenance of the journal in ISO alpha-2 format
- The subject field in the LCC classification and we divide them into Social Science and Humanities (SSH) and Science Technology and Mathematics (STM).

```
py -m cleaner doaj_article_data_[date generated] #macos
python3 -m cleaner doaj_article_data_[date generated] #macos

```

Result example

```
{
    "XXXX-XXXX": {
        "code": "XXXX", #eissn or pissn
        "country": "XX", country code alpha 2
        "subject": [
            {
                "code": "X",
                "scheme": "XXX",
                "term": "some_term"
            },
            {
                "code": "X",
                "scheme": "XXX",
                "term": "some_term"
            },
            {
                "code": "X",
                "scheme": "XXX",
                "term": "some_term"
            }
        ]
    },
...
```

When dealing with subject data, we started from the **subject codes** provided by Crossref, selecting the first item in the subject's array item ['subject']['code'] as a value for this additional column.

Example node:

"subject": [{"code": "L", "scheme": "LCC", "term": "Education"}

Then subjects are divided into two macro groups:

- Science, Technology and Medicine (STM)
- Social Science and Humanities (SSH)

5 Populate the final dataset with **populator.py**.
This script uses information from the journals and adds it to the articles.
With this script, we added information about the country of origin of the journal and the subject of the journal.

command to launch it from the terminal:

```
py -m populator path
```

Results template:

```
[‘issn', 'doi', 'doi-num', 'on-crossref', 'reference', 'asserted-
by-cr', 'asserted-by-pub', 'ref-undefined', 'ref-num', 'year',
'country', 'subject']
```

Finally, we created a pickle file to have the dataframe in one place simplifying the following processes.

Data Analysis & Visualisation

6 Starting from our final dataframe, [20220521_la_chouffe_aggregated_data_v_0_0_1.pkl], we loaded it on our pyton using the commands:

```
df.describe #for describing the whole df
------------
df[df["column-name"] == value].describe() #for describing a subset
of the df
```

For getting statistical data on the df columns, such as numerical values means, counts

With our research we used multiple measures that we repeated in the different scopes of our research, in particular:

RQ1: How many articles published in the open access journals in DOAJ are included in Crossref?
RQ2: How many of these articles include information about their reference lists?
RQ3: How many references have a DOI specified?
RQ4: How many of these DOIs have been specified by the publishers? And how many by Crossref?

6.1 As we printed the description results for the whole dataset and its subgroup having 'reference' == 1, we decided to first look for a general description of the variables involved in answering each one of our research questions.

Here, we selected the first kind of data visualization we wanted to look at, to get a general understanding of what is happening in the OA field, regarding our study objectives.

Plotly is the python library used at this stage.

To answer **RQ1**, we present the percentage of articles also presented on Crossref over the total number of articles in DOAJ.
The data used is:

```
totdoi = len(working) #tot number of dois over our
dataframe
```

```
oncross = working['on-crossref'].sum() # tot DOIS also
on Crossref
notOn = totdoi - oncross # tot DOIS not on crossref
```

To answer **RQ2**, the percentage of DOAJ articles on Crossref having a reference list over the tot amount of DOAJ articles present there was plotted.
The data used is:

```
working = working[working['type'].isin(['journal-
article', 'book', 'book-chapter', 'proceedings-article',
'dataset', 'posted-content', 'report'])]

noRef = len(working) -working['reference'].sum()#DJ arts
on Cr w/out ref
totdoi = working['reference'].sum() # tot DOAJ articles
on Cr having references
```

To answer **RQ3**, the percentage of DOAJ articles on Crossref having a reference list over the tot amount of DOAJ articles present there was plotted.
The data used is:

```
ref_defined = working['ref-num']-working['ref-
undefined'] # ref defined by someone
ref_defined = ref_defined.sum() # tot references w/ DOI
defined by someone
ref_undefined = working['ref-undefined'].sum() # tot
references w/out DOI
```

To answer **RQ4**, we present the percentages of DOAJ articles having: references' DOIs asserted by Crossref, references' DOIs asserted by Publisher, and no references' DOIS specified at all.
The data used is:

```
ass_cross = working['asserted-by-cr'].sum() #tot num of
DOIs asserted by Crossref
ass_pub = working['asserted-by-pub'].sum()  #tot num of
DOIs asserted by publishers
und = working['ref-undefined'].sum()  #tot num of DOIs
not asserted
```

After setting these counts as the values we wanted to focus on, we made a pie-chart for each question following the templates you can find in our notebook on Data Analysis.

Here we took a break from our code.
This hands-off time was invested in making some observations on the

new insights into our data we got thanks to **data_viz.ipynb**!

We recommend you do the same here!

6.2 Further Inquiry

As we moved towards a more granular description of our data we started introducing additional variables such as country and subject – i.e., the research field – by doing this we also introduce new kinds of visualizations to be later selected, depending on the relevance of the information we could infer from our data evidence.

We start re-iterating all of our research questions.

The first variable introduced in this new iteration is the subject of the OA journal the articles belong to, then we also introduce the country and finally explore the trend over time of the percentage of DOAJ dois on Crossref

RQ1

1. Violin-plot

This visualization was selected to show the distribution of the DOAJ articles registered on Crossref, which we presented in the first pie chart.

First, we considered only doi published between 1950 and 2022, then we grouped the frame by Issn:

```
frame = frame[(frame.year >= 1950)&(frame.year < 2022)]
frame = frame.groupby('issn').sum()
```

- Y axis: percentage of dois present on Crossref

```
frame['perc_cr'] = (frame['on-crossref']/frame['doi-num'])*100 # Y axis data (floats)
```

2. Bar-plot by subject

In this second graph, we grouped the working df by 'subject' and summed the numerical values in it

```
frame = frame.groupby('subject').sum()
```

- X axis: subjects of DOAJ journals
- Y axis: percentage of dois present on Crossref

```
frame['subject'] # Y axis data (categorical)
frame['perc_cr'] = (frame['on-crossref']/frame['doi-
num'])*100 # X axis data (floats)
```

3. Bar-plot by country

In this bar-plot we groupped the working df by 'country' and summed the numerical values in it. Then, we filtered out rows (i.e. countries) having percentages of dois on Crossref greater than 0 and lower than 80%.

```
#import library that changes the country codes (alpha-3)
into names
import country_converter as coco
import pycountry_convert as pc

frame = frame.groupby('country').sum()
frame['country-name'] = coco.convert(names=frame.index,
to="name")
frame = frame[(frame.perc_cr < 80) & (frame.perc_cr >
0)]
```

We add the continent names column

```
for x in frame3.index:
    country_continent_code =
pc.country_alpha2_to_continent_code(x)
    country_continent_name =
pc.convert_continent_code_to_continent_name(country_cont
inent_code)
    continent_name.append(country_continent_name)
frame['continent'] = continent_name
```

- X axis: countries of DOAJ journals
- Y axis: percentage of dois present on Crossref
- Colour scale: continent

```
frame['country-name'] # Y axis data (categorical)
frame['perc_cr'] = (frame['on-crossref']/frame['doi-
num'])*100 # X axis data (floats)
frame['continent'] # Color scale data (categorical)
```

RQ2

1. Violin-plot

This visualization was selected to show the distribution of the articles having metadata on references, over the total number of DOAJ's articles listed on Crossref .

First, we considered only doi published between 1950 and 2022, then we grouped the frame by Issn:

```
frame = frame[(frame.year >= 1950)&(frame.year < 2022)]
frame = frame.groupby('issn').sum()
```

- Y axis: percentage of dois present on Crossref having a reference list

```
frame['perc_ref'] = (frame['reference']/frame['on-
crossref'])*100 # Y axis data (floats)
```

2. Bar-chart – by subject

In this chart, we present the data frame divided into subjects, each described by the percentage of references list of their articles on Crossref, and by the total count of references

We grouped the working df by 'subject' and summed the numerical values in it.

```
frame = frame.groupby('subject').sum()
```

- X axis: subjects of DOAJ journals
- Y axis: percentage of references present on Crossref

```
frame['subject'] # X axis data (categorical)
frame['perc_ref']= (frame['reference']/frame['on-
crossref'])*100 # Y axis data (floats)
```

3. Bar-chart – by country/continent

In this chart, we present the data frame divided into countries, each described by the percentage of references list of their articles on Crossref, and by the total count of references

We grouped the working df by 'subject' and summed the numerical values in it.

```
frame = frame.groupby('country').sum()
frame['perc_ref'] = (frame['reference']/frame['on-
crossref'])*100
frame['country-name'] = coco.convert(names=frame.index,
to="name")
```

- X axis: countries of DOAJ journals
- Y axis: percentage of references present on Crossref
- Colour scale: Tot number of references

```
frame['subject'] # X axis data (categorical)
frame['perc_ref']= (frame['reference']/frame['on-
crossref'])*100 # Y axis data (floats)
frame['ref-num'] # Color scale data (int)
```

4. **Line-chart – by year**

This visualization is characterized by 2 traces having the purpose of comparing the percentages of both DOAJ dois and their reference lists on Crossref, again on the year span 1950-2022.

```
frame = frame[(frame.year >= 1950)&(frame.year < 2022)]
frame = frame.groupby('year').sum()
```

- X axis: pub year of DOAJ articles
- Y axis: percentage dois and reference lists on Crossref

```
frame['perc_ref'] = (frame['reference']/frame['on-
crossref'])*100
frame['perc_cr'] = (frame['on-crossref']/frame['doi-
num'])*100
```

```
frame['year'] # X axis data (ordinal)
frame['perc_ref'] # Y axis first trace data (floats)
frame['perc_cr'] # Y axis second trace data (floats)
```

RQ3

1. **Stacked bar-chart – by subject**

This also doubles traced visualization compares the number of references having specified DOIs and those not having one defined over subjects.

```
frame = frame.groupby('subject').sum()
```

We calculate the needed percentages

```
frame['perc_ref_nodoi'] = (frame['ref-
undefined']/frame['ref-num'])*100
frame['perc_ref_doi'] = 100 - frame['perc_ref_nodoi']
```

- X axis: subjects of DOAJ articles
- Y axis: percentage of DOIs specified and percentage of DOIs not specified

```
frame['subject'] # X axis data (categorical)
frame['perc_ref_nodoi'] # Y axis first trace data
(floats)
frame['perc_ref_doi'] # Y axis first trace data (floats)
```

2. **Stacked bar-chart – by country**

This also doubles traced visualization compares the number of references specified DOIs or not specified at all, focusing on the country.

First here, subset the working frame by country, compute ref_precentages as above and change our iso-alpha 2 codes into iso-alpha 3

```
frame22['iso_alpha'] = coco.convert(names=frame22.index,
to='ISO3')
frame22['country-name'] =
coco.convert(names=frame22.index, to='name')
frame22['perc_ref_nodoi'] = (frame22['ref-
undefined']/frame22['ref-num'])*100
frame22['perc_ref_doi'] = 100 -
frame22['perc_ref_nodoi']
```

- Locations: countries of DOAJ as iso-alpha 3 codes
- Colour: percentage of references with DOIs on Crossref and percentage of references without DOIs on Crossref

```
frame22['iso_alpha'] # Locations data (categorical)
frame22['perc_ref_nodoi'] # Color first trace data
(float)
frame22['perc_ref_doi'] # Color first trace data (float)
```

RQ4

1. **Stacked bar-chart – by subject**

This also doubles traced visualization compares the number of references having specified DOIs by crossref, publisher or not specified at all, focusing on the subjects.

```
frame26 = frame26.groupby('subject').sum()
```

We calculate the needed percentages

```
frame26['perc_asserted_cr'] = (frame26['asserted-by-
cr']/frame26['ref-num'])*100
frame26['perc_asserted_pub'] = (frame26['asserted-by-
pub']/frame26['ref-num'])*100
frame26['perc_ref_nodoi'] = (frame26['ref-
undefined']/frame26['ref-num'])*100
```

- X axis: subjects of DOAJ articles
- Y axis: percentage of DOIs specified by crossref, publisher or not specified at all

```
frame26['subject'] # X axis data (categorical)
frame26['perc_asserted_cr'] # Y axis first trace data
(floats)
frame26['perc_ref_nodoi'] # Y axis first trace data
(floats)
frame26['perc_ref_doi'] # Y axis first trace data
(floats)
```

2. **Stacked bar-chart – by country**

This also doubles traced visualization compares the number of references having specified DOIs by crossref, publisher or not specified at all, focusing on the country.

```
frame27 = frame27.groupby('subject').sum()
```

We calculate the needed percentages

```
frame27['perc_asserted_cr'] = (frame27['asserted-by-
cr']/frame27['ref-num'])*100
frame27['perc_asserted_pub'] = (frame27['asserted-by-
pub']/frame27['ref-num'])*100
frame27['perc_ref_nodoi'] = (frame27['ref-
undefined']/frame27['ref-num'])*100
```

- X axis: country of DOAJ articles
- Y axis: percentage of DOIs specified and percentage of DOIs not specified

```
frame27['subject'] # X axis data (categorical)
frame27['perc_asserted_cr'] # Y axis first trace data
(floats)
frame27['perc_ref_nodoi'] # Y axis first trace data
(floats)
frame27['perc_ref_doi'] # Y axis first trace data
(floats)
```

3. **Histogram – by year**

With this visualization, we compare the trends of references s the number of references having specified DOIs by crossref, publisher or not specified at all over the time range 1950-2022

```
frame28 = frame28[(frame28.year >= 1950)&(frame28.year <
2022)]
frame28 = frame28.groupby('year').sum()
```

- X axis: years of DOAJ articles
- Y axis and colour: percentage of references having specified DOIs by crossref, publisher or not specified at all

```
frame28['year'] # X axis data (ordinal)
frame28['perc_asserted_cr'] # Y axis first trace data
(floats)
frame28['perc_ref_nodoi'] # Y axis first trace data
(floats)
frame28['perc_ref_doi'] # Y axis first trace data
(floats)
```

Publishing Data

7 Software, dataset, and metadata publication

- **Metadata and datasets** have been published in compressed formats on Zenodo, the software can be found on GitHub at this address: https://github.com/open-sci/2021-2022-la-chouffe-code and on Zenodo
- 20220521_la_chouffe_clean_articles_v_0_0_1.tar.gz and

20220521_la_chouffe_articles_populated_v_0_0_1.tar are both in permissive licences, CC0.

- 20220521_la_chouffe_aggregated_data_v_0_0_1.pkl has a Cretive Common ShareAlike licence 4.0.
- The software created has an MIT licence.