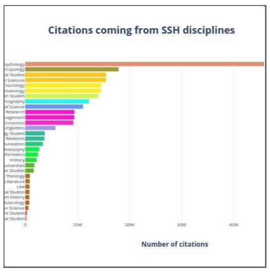


VERSION 4

SEP 07, 2023

DOI:
dx.doi.org/10.17504/protocols.io.n92ldpeenl5b/v4

Protocol Citation: Marta Soricetti, Sara Vellone, Olga Pagnotta, Lorenzo Paolini 2023. Uncovering the Citation Landscape: Exploring OpenCitations COCI, OpenCitations Meta, and ERIH-PLUS in Social Sciences and Humanities Journals - Workflow. **protocols.io** https://dx.doi.org/10.17504/protocols.io.n92ldpeenl5b/v4Version created by Olga Pagnotta

Protocol status: Working
We use this protocol and it's working

Created: Sep 07, 2023

Last Modified: Sep 07, 2023

PROTOCOL integer ID: 87489

Keywords: OpenScience, Citation, OC-COCI, OC-Meta, ERIH-PLUS, journals

Uncovering the Citation Landscape: Exploring OpenCitations COCI, OpenCitations Meta, and ERIH-PLUS in Social Sciences and Humanities Journals - Workflow V.4

Marta Soricetti[1], Sara Vellone[1], Olga Pagnotta[1], Lorenzo Paolini[1]

[1]University of Bologna

Pika.py Pika.py

Sara Vellone

ABSTRACT

Purpose

The main purpose of this research is to answer to three different questions and find out:

- the number of citations which refer to publications in Social Sciences and Humanities journals included in ERIH-PLUS, by looking at citations data contained in OpenCitations COCI and OpenCitations Meta;
- the most citing and the most cited SSH discipline, according to the above mentioned datasets;
- the citations coming from and going to publications contained in OpenCitations Meta which are not included in SSH journals. We want to draw a line that connects these three different datasets, aiming at offering an overall view of the citations landscape of each of them.

Methodology

For this purpose, we approach the problem from a computational point of view. We extract only the relevant data by operating a first preprocessing of COCI, ERIH-PLUS and META's datasets. Then we build a python software able to analyze CSVs data, querying them to retrieve information needed and to present the results in a clear and understandable way.

Findings

The findings show that the majority of citations come from and go to psychology publications, and a deep gap exists between the number of citations included in SSH journals and the number of citations that are not included in SSH journals.

Originality/Value

The research conducted by us has the purpose to add information to existing resources with the aim of facilitating their use and allowing the researchers to have a clearer view of the data contained in each dataset. In addition, the research has the purpose to gather information that may be useful for understanding which is the most influential discipline in the SSH field and to provide a solid starting point for further studies regarding this subject.

Keywords: COCI, Meta, ERIH-PLUS, OpenCitation, SSH, Journals

GUIDELINES

The required Python version for running the current software is Python 3.10.
Other libraries needed are:

- numpy==1.24.3
- pandas==2.0.1
- python-dateutil==2.8.2
- pytz==2023.3
- six==1.16.0
- tqdm==4.65.0
- tzdata==2023.3
- zstandard==0.21.0

The library CSVManager needs to be manually installed by downloading the corresponding folder from the linked Github repository.

MATERIALS

DMP of the research: https://doi.org/10.5281/zenodo.8324973
Article of the research: https://zenodo.org/record/8324989
Software of the research: https://doi.org/10.5281/zenodo.8324961
Datasets of the research: https://doi.org/10.5281/zenodo.797315, https://doi.org/10.5281/zenodo.7974816

Data description and analysis

1 The datasets that we used in our research are the following:

- **OpenCitations COCI**[1]: COCI dump
- **OpenCitations Meta**[2]: Meta dump
- **ERIH-PLUS**[3]: list of approved journals

Dataset	
COCI	NAME
https://opencitations.net/download#coci	LINK

Dataset	
META	NAME
https://opencitations.net/download#meta	LINK

Dataset	
ERIH-PLUS (approved journals)	NAME
https://kanalregister.hkdir.no/publiseringskanaler/erihplus/periodical/listApproved	LINK

[1]v19 (released on January 2023)
[2]v3 (released on February 2023)
[3]version downloaded on 2023-04-27

1.1 ***COCI, the OpenCitations Index of Crossref open DOI-to-DOI citations***

COCI is an RDF dataset containing details of all the citations that are specified by the open references to DOI-identified works present in Crossref.
It includes, in this latest version: 1,463,920,523 citations and 77,045,952 bibliographic resources.

The following are the names of the column of the dataset:

- **[field "oci"]**the Open Citation Identifier (OCI) for the citation;
- **[field "citing"]**the DOI of the citing entity;
- **[field "cited"]**the DOI of the cited entity;
- **[field "creation"]**the creation date of the citation (i.e. the publication date of the citing entity);
- **[field "timespan"]**the time span of the citation (i.e. the interval between the publication date of the cited entity and the publication date of the citing entity);
- **[field "journal_sc"]**it records whether the citation is a journal self-citations (i.e. the citing and the cited entities are published in the same journal);
- **[field "author_sc"]**it records whether the citation is an author self-citation (i.e. the citing and the cited entities have at least one author in common).

oci	citing	cited	creation	timespan	journal_sc	author_sc
02001000106361937271423142314370200020137010237010303-020010001063619371614242917142722181228370200010737000437000004	10.1016/j.renene.2021.12.133	10.1016/j.geothermics.2017.04.004	2022-03	P4Y6M	no	no

1.2 *META*

OpenCitations Meta stores and delivers bibliographic metadata for all publications involved in the OpenCitations Indexes.
It includes, in this latest version: 90,102,757 bibliographic entities; 282,247,615 authors and 2,367,265 editors; 644,830 publication venues and 18,397 publishers.

The following are the names of the column of the dataset:

- **[field "id"]**the IDs for the document described within the line;
- **[field "title"]**the document's title;
- **[field "author"]**the authors of the document;
- **[field "pub_date"]**the date of publication;
- **[field "venue"]**information about the venue, i.e. the bibliographical resource to which the document belongs;
- **[field "volume"]**the volume sequence identifier (e.g. a number) to which the entity belongs;
- **[field "issue"]**the issue sequence identifier (e.g. a number) to which the entity belongs;
- **[field "page"]**the page range of the resource described in the row;
- **[field "type"]**the type of resource described in the row;
- **[field "publisher"]**the entity responsible for making the resource available;
- **[field "editor"]**the editors of the document.

id	title	author	issue	volume	venue	page	pub_date	type	publisher	editor
"meta:br/060209 doi:10.4230/lipics.approx/random.2020.19"	"Distributed Testing Of Graph Isomorphism In The CONGEST Model"	"Levi, Reut [meta:ra/0610110096 orcid:0000-0003-3167-1766]; Medina, Moti [meta:ra/0612046435 orcid:0000-0002-5572-3754]"	""	""	" [meta:br/060182 issn:1868-8969]"	""	"2020"	"report"	"Schloss Dagstuhl - Leibniz-Zentrum Für Informatik [meta:ra/0605251]"	"Byrka, Jarosław [meta:ra/069044096 orcid:0000-0002-3387-0913]; Raghu Meka [meta:ra/0605252]"

1.3 *ERIH-PLUS (list of approved journals)*

ERIH PLUS is an academic journal index for the HSS (Humanities and Social Sciences) society in Europe.
This version (27/4/23) contains 11128 records.

The following are the names of the columns of the dataset:

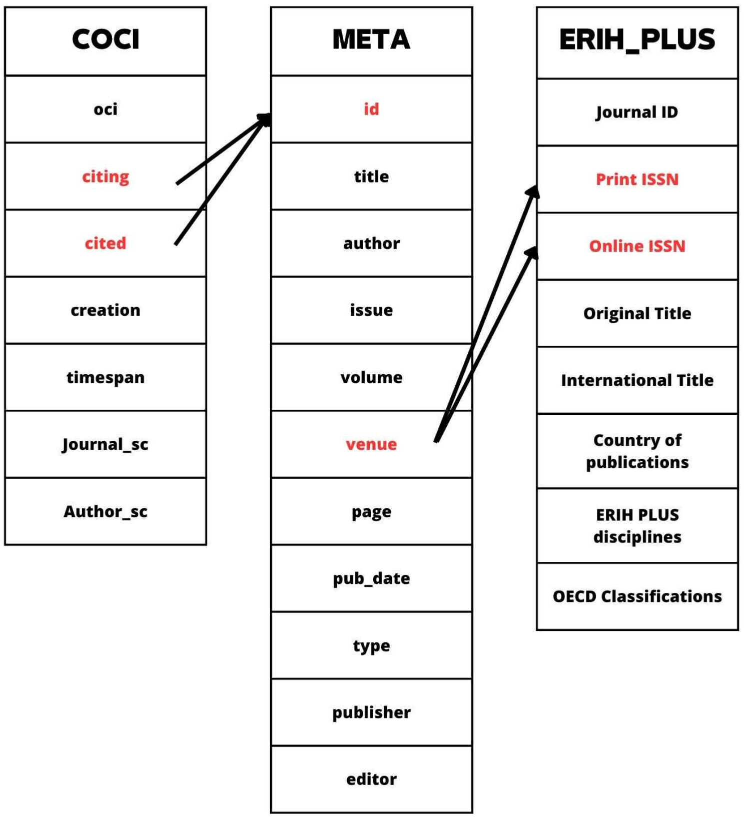
- **Journal ID**
- **Print ISSN**
- **Online ISSN**
- **Original Title**
- **International Title**
- **Country of Publication**
- **ERIH PLUS Disciplines**
- **OECD Classifications**
- **[Last Updated],** which contains data about the last update of each journal in the dataset

Journal ID	Print ISSN	Online ISSN	Original Title	International Title	Country of Publication	ERIH PLUS Disciplines	OECD Classifications	[Last Updated]
488138	1392-4095	2351-6526	Acta Historica Universitatis Klaipedensis	Acta Historica Universitatis Klaipedensis	Lithuania	History	History and Archaeology	02/02/2023 17:14:12

Unfortunately a detailed description of the data contained in each column is not provided.

Processing of Input Data

2 We tried to define a mapping of the datasets to understand what are the information that the three datasets have in common. By looking at the data and the columns' names, we have identified the following:



Taking as a starting point META, we have identified the COCI columns "citing" and "cited" as corresponding to DOIs, which are contained also in the "id" column of META.
For what concerns EIRH-PLUS, we have noticed that "print ISSN" and "online ISSN" correspond to the "venue" column of META.

We have processed the data by filtering and cleaning them, keeping only the relevant information for the research purpose.

Preprocessing
For META and COCI we have reused some methods of the OpenCitations Preprocessing software

Software

Preprocess	NAME
OpenCitations	DEVELOPER

adapting it to our needs.

For the preprocessing of META and COCI datasets we have created a Superclass, called **Preprocessing**, which contains the method **"get_all_files"**.
The method retrieves files based on the provided input directory or compressed file (i_dir_or_compr) and the requested file type (req_type). It

initializes an empty list named result and a variable named targz_fd, set to None. The method checks the type of i_dir_or_compr:

- If it is a directory (isdir(i_dir_or_compr)), it enters the directory using walk and appends the files that match the requested type and do not start with a dot (.) to the result list.
- If it ends with the extension "tar.gz", it opens the file using tarfile.open and iterates over the files within it. It appends the files that match the requested type and do not start with a dot (.) to the result list.
- If it ends with the extension "zip", it extracts the contents of the ZIP file to a destination directory. Then it traverses the destination directory using walk and appends the files that match the requested type and do not start with a dot (.) to the result list.
- If it ends with the extension "zst", it decompresses the Zstandard-compressed file to a destination directory. Then it enters the destination directory using walk and appends the files that match the requested type and do not start with a dot (.) to the result list.
- If it ends with the extension ".tar", it extracts the contents of the TAR file to a destination directory. Then it traverses the destination directory using walk and appends the files that match the requested type and do not start with a dot (.) to the result list.
- If none of the above conditions are met, it prints a message stating that it's not possible to process the input path.

Finally, the method returns the result list of file paths and the targz_fd object. Note that there are dependencies on external libraries such as tarfile, zipfile and zstd that are used for handling different compression formats.

MetaPreProcessing and **CociPreProcessing** inherit from the Preprocessing class.
These classes share the method **splitted_to_file**, adapted to the structure of the output files produced by each class, and the method **split_input**, which is in both the classes the main function for reading, filtering and storing the processed versions of the data in the output files.
Splitted_to_file
Let's describe the functioning of splitted_to_file. The method takes four parameters: cur_n, lines, columns_to_use and output_dir_path.

- The method first checks if the integer value of cur_n is not zero (int(cur_n) != 0) and if it is divisible by the integer value of self._interval (int(cur_n) % int(self._interval) == 0), a parameter used for instantiating the class that controls the number of lines to store in each output file.
- If the condition is true, it proceeds with the following steps to create a new file.
- If the condition is false, it skips the file creation and returns the lines unchanged.

If the condition is true:

- It generates a filename based on the value of cur_n and self._interval by concatenating "filtered_", the result of cur_n divided by self._interval, and self._req_type.
- If a file with the generated filename already exists in the output_dir_path, it appends the current date and time to the filename to make it unique.
- It opens the file for writing using open with the appropriate encoding and newline options.
- It creates a csv.DictWriter object named dict_writer with the file handle, specifying the delimiter as ",", quoting style as csv.QUOTE_ALL, escape character as "\" and the field names as columns_to_use.
- It writes the header row to the file using dict_writer.writeheader().
- It writes all the rows in lines to the file using dict_writer.writerows(lines).
- It closes the file handle using f_out.close().
- It clears the lines list by assigning an empty list to it (lines = []).
- It returns the empty lines list.

If the condition is false:

- It simply returns the lines list unchanged.

The method is overall responsible for splitting a list of lines into separate files based on a specified interval. The interval is defined by the self._interval
value, which is an attribute of the class instance, as specified above. Each file is created with a filename that includes a prefix, a number indicating the split index, and the file extension self._req_type. The files are created in the specified output_dir_path directory.

Split_input
This is the method that orchestrates the entire processing in both the classes: it iterates over the input files, reads the data, filters and processes them and calls the splitted_to_file method to write the new output files. The processing part is different in the two classes, but in both of them, the reading of the data is managed with the *zipfile* library (in particular the methods ZipFile, namelist and open) because the original datasets (COCI and META) are formed by multiple CSV files stored in zip files. It then uses the *pandas* library (pd) to read the CSV files in chunks of 10000 rows and converts each chunk into a list of dictionaries for further processing, described in details further in the present protocol.

2.1 ***META***

META is our focus for answering the research questions, but we have performed some filtering also on this dataset to be able to merge it with the others.
In the class **MetaPreProcessing** we manage the processing of the META dump.
For the columns "id" and "venue" of the original files we have decided to keep as identifiers of publications and venues only, respectively, the DOIs and the ISSNs, removing thus all the other identifiers specified for each entity in META, obtaining the following structure:

id	title	author	issue	volume	venue	page	pub_date	type	publisher	editor
"doi:10.4230/lipics.approx/random.2020.19"	"Distributed Testing Of Graph Isomorphism In The CONGEST Model"	"Levi, Reut [meta:ra/0610110096 orcid:0000-0003-3167-1766]; Medina, Moti [meta:ra/0612046435 orcid:0000-0002-5572-3754]"	""	""	"issn:1868-8969"	""	"2020"	"report"	"Schloss Dagstuhl - Leibniz-Zentrum Für Informatik [meta:ra/0605251]"	"Byrka, Jarosław [meta:ra/069044096 orcid:0000-0002-3387-0913]; Raghu Meka [meta:ra/0605252]"

the data in bold are those we have filtered

Split_input
As said before, **split_input** is the method in charge of the entire processing that now will be described in details. The preprocessing is made by iterating over each dictionary line in the list of dictionaries obtained from the chunks of the CSVs files (section 2), and incrementing the count variable initialized at the begininning (count = 0) by 1.
The method goes on with the following steps:

- It splits the value of the "id" key in the line dictionary into a list of identifiers ids_list using line.get("id").split(" ").
- It initializes an empty string new_doi_key to store the valid DOI identifiers.
- It iterates over each identifier id in the ids_list (in the META "id" columns there are different identifiers).
- It checks if the id matches the regular expression "^doi:10\.(\d{4,9}|[^\s/]+(\.[^\s/]+)*)/[^\s]+$"

using match("^doi:10\.(\d{4,9}|[^\s/]+(\.[^\s/]+)*)/[^\s]+$", id).

- It removes the trailing space from the new_doi_key using sub("\s$", "", new_doi_key).
- If the line dictionary has a key "venue" and its value is not empty, it further processes the venue identifiers.
- It removes the square brackets "[" and "]" from the "venue" value using sub("[", "", line.get("venue")) and sub("]", "", line.get("venue")).
- It splits the modified "venue" value into a list of venue identifiers venue_ids_list using venue_ids_list.split(" ").
- It initializes an empty string new_issn_key to store the valid ISSN identifiers.
- It iterates over each identifier venue_id in the venue_ids_list.
- It checks if the venue_id matches the regular expression pattern "^issn:[0-9]{4}-[0-9]{3}[0-9X]$" using match("^issn:[0-9]{4}-[0-9]{3}[0-9X]$", venue_id). If it matches, it appends the venue_id to the new_issn_key string with a space.
- It removes the trailing space from the new_issn_key using sub("\s$", "", new_issn_key).
- If the "id" key in the line dictionary is not an empty string (indicating it has a valid DOI), it appends the line dictionary to the lines list (initialized as empty at the beginning).
- If the count is not zero and is divisible by self._interval, it calls the **splitted_to_file** method to write the accumulated lines to a file.
- After processing all chunks and dictionaries, if there are any remaining lines in the lines list, it adjusts the count by adding the difference

Create_list_dois
The **MetaPreProcessing** class has also another method, **create_list_dois**, that is responsible for creating CSV files containing the DOIs of all the publications stored in META (**list_meta_dois**).
Two variables, lines and count, are initialized. Lines is an empty list that will store filtered entities to be saved in output files, while count keeps track of the number of rows added to the final CSVs. The method calls **get_all_files** (described in section 2), with two arguments: the path of the directory where is stored **META_preprocessed** and self._req_type. The purpose of the method is to retrieve a list of files and a file descriptor.
During the files iteration, it uses the *pandas* library (pd) to read the CSV files in chunks of 10000 rows and converts each chunk into a list of dictionaries, where each dictionary represents a row. The method then iterates over each dictionary (line) in the df_dict_list. It splits the value of the 'id' key (META infact contains different IDs in the same cell) by a space delimiter to obtain a list of IDs (ids_list). It further iterates over each ID, creates a new dictionary new_line with the 'id' key and the current ID value, appends it to the lines list, and increments the count variable.
After each ID is processed, the method checks if count is not zero and if count is divisible by self._interval without any remainder. If this condition is met, it calls the method, **splitted_to_file**, with the appropriate arguments to split and save the accumulated lines list into an output file.

Info about META_preprocessed:
- 8438 files with self._interval = 10000;
- 32.2 GB unzipped

Info about list_meta_dois:
- 7623 files with self._interval = 10000;
- 2.51 GB unzipped

2.2 *COCI*

In the class **CociPreProcessing** we manage the preprocessing of the COCI dump.
After the preprocessing, we will keep only the citations that are entirely contained in META. This means that the citations which have either the citing or the cited entity (or both) not contained in META are excluded from COCI_preprocessed. The method checks this using the files produced by MetaPreProcessing containing all the DOIs of META (that are passed as input of the class). The output files will be thus formed by two columns, "citing" and "cited". The method of the class in charge of the preprocessing of COCI is **split_input**.

Split_input
In this class, the method takes as input also a boolean parameter, list_dois_excluded_from_meta, that controls the creation of additional files (**excluded_dois_from_meta**) giving information about the citations that involve publications not in META.
At the beginning of the method several variables are initialized. lines_coci_pre is an empty list to store filtered entities related to COCI preprocessing, count keeps track of the number of iterations for COCI preprocessing, and lines_dois_excluded and count_dois_excluded are initialized only if list_dois_excluded_from_meta is true.
The files produced by **MetaPreProcessing** containing all the DOIs of META are loaded as a set through the method **load_csv_column_as_set** of the class **CSVManager** (set_meta_id = CSVManager.load_csv_column_as_set(self._list_meta_dois_path, "id"); the link to the original OpenCitations class can be found in section 3.1).
The preprocessing is made by iterating over each dictionary (line) in the list of dictionaries obtained from the chunks of the CSVs files (section 2).
It retrieves the values of the "citing" and "cited" keys (doi_citing and doi_cited). It checks if the citing and cited DOIs are present in the set_meta_id set, and based on that, sets boolean variables (citing_in_meta, cited_in_meta, citing_not_in_meta, cited_not_in_meta) accordingly:
- If both the citing and cited DOIs are present in the set_meta_id set, it increments the count variable, modifies the "citing" and "cited" values in the line dictionary to include the "doi:" prefix, appends the modified line to lines_coci_pre, and if the count is divisible by self._interval, it calls the splitted_to_file method to split and save the accumulated lines_coci_pre into an output file.
- If list_dois_excluded_from_meta is true and either the citing or cited DOIs are not present in the set_meta_id set, it increments the count_dois_excluded variable and creates a new dictionary (new_line) based on the presence of the DOIs in the set. It appends the new_line to lines_dois_excluded. Similarly, if the count is divisible by self._interval, it calls the **splitted_to_file** method to split and save the accumulated lines_dois_excluded into another output file.

After the iterations, the method checks if lines_coci_pre has any elements. If it does, it adjusts the count variable to the next multiple of self._interval and calls the **splitted_to_file** method to save the remaining lines_coci_pre into an output file. If list_dois_excluded_from_meta is true and lines_dois_excluded has any elements, it adjusts the count_dois_excluded variable to the next multiple of self._interval and calls the **splitted_to_file** method to save the remaining lines_dois_excluded into another output file.

"citing"	"cited"
"doi:10.1016/j.renene.2021.12.133"	"doi:10.1016/j.geothermics.2017.04.004"

example of the dataset COCI_preprocessed

"citing"	"is_citing_in_meta"	"cited"	"is_cited_in_meta"
"doi:10.1007/978-1-137-49092-6_5"	"False"	"doi:10.1016/0010-440x(73)90041-2"	"False"

example of the dataset resulting from the method split_input setting list_dois_excluded_from_meta equal to True

Info about COCI_preprocessed:
- 13967 files with self._interval = 100000;
- 90.76 GB unzipped

Info about excluded_dois_from_meta:
- 673 files with self._interval = 100000;
- 5.43 GB

2.3 *ERIH_PLUS*

The class **ErihPreProcessing** is responsible for the preprocessing of ERIH_PLUS dataset. It creates a new CSV file with two columns "venue_id" and "ERIH_disciplines". "venue_id" is the union of the original columns "Online ISSN" and "Print ISSN" of ERIH_PLUS.
This class is different from **CociPreProcessing** and **MetaPreProcessing** mainly for the reason that the ERIH_PLUS dataset is smaller than COCI and META.

Preprocess_ERIH_plus
The method **preprocess_ERIH_plus** is the main method of the class, the one that orchestrates the preprocessing.
It reads the CSV file using open and csv.reader, skipping the first line.
For each row in the CSV file, it creates two dictionaries (venue_dict and venue_dict2), to store values coming from the columns "Print ISSN" and "Online ISSN" of ERIH_PLUS.
If both columns row[1] ("Print ISSN") and row[2] ("Online ISSN") have values, it adds the "venue_id" key to both dictionaries having as values the columns' contents annotated with the prefix "issn:".
If in a row, just one of the columns "Print ISSN" and "Online ISSN" is filled, the corresponding value is added. If the column "ERIH PLUS Disciplines" is not empty, it adds an "ERIH_disciplines" key to both dictionaries with the corresponding value.
The two dictionaries, venue_dict and venue_dict2 (if not empty), are appended to ERIH_preprocessed list, that is returned as output of the method.

The other class method, **write_csv**, writes the preprocessed data to a new CSV file ("erih_preprocessed.csv").

"venue_id"	"ERIH_disciplines"
"issn:1392-4095 issn:2351-6526"	History

Info about ERIH_preprocessed:
- 1 file;
- 1.3 MB

Further processing of Data

3 The previous steps allowed us to create three cleaned datasets, each composed by different CSV files: **META_preprocessed**, **COCI_preprocessed**, **ERIH_preprocessed**. To answer the research questions we have performed some further operations upon META and ERIH: we have merged **META_preprocessed** dataset together with **ERIH_preprocessed**, obtaining thus new CSVs having as columns all the columns of META with the addition of the ERIH-PLUS disciplines.

3.1 *ERIH_META*

The merged dataset obtained with **META_preprocessed** and **ERIH_preprocessed** has been obtained by creating a new class, **ErihMeta**, which has four internal methods.
The method **get_all_files** retrieves all the files in the specified directory (i_dir_or_compr) that match the specified file extension (req_type). It returns a list of file paths.
The method **splitted_to_files** has the same structure as the one described in section 2.
The method **find_erih_venue** returns the corresponding ERIH-PLUS disciplines given an input ISSN list, composed of ISSN taken from META_preprocessed column "venue". Within this method we have used the class **CSVManager** of OpenCitations

Software

OCMeta, CSVManager NAME

https://github.com/opencitations/oc_meta SOURCE LINK

and in particular the method **get_value**, adapting the whole class to the structure of ERIH_preprocessed dataset.
The method **erih_meta** is the main method of the class, it is responsible for the actual processing of the data, which will be explained below.

Find_erih_venue
The method **find_erih_venue** takes a list of ISSN (International Standard Serial Number) values as input and returns a string representing the ERIH disciplines associated with those ISSN values. It uses the _erih_preprocessed_path object (the path of the folder where is stored ERIH_preprocessed required for instantiating the class) to retrieve the discipline information.

Erih_meta
erih_meta is the main method, as said before, that performs the **ERIH_META** processing. It reads the preprocessed META files, iterates through the chunks of data, and processes each line. It extracts the venue information and calls the **find_erih_venue** method to determine the ERIH disciplines associated with the venue (if there is not any discipline associated to a venue ISSN, the column "erih_disciplines" will be filled with an empty string). The resulting processed lines are appended to the lines list. If the count reaches the interval value, the lines list is written to a CSV file using the **splitted_to_file** method. Finally, if there are any remaining lines in the lines list, a final file is created to store them.

"id"	"title"	"author"	"issue"	"volume"	"venue"	"page"	"pub_date"	"type"	"publisher"	"editor"	"erih_disciplines"
"doi:10.1207/s15327078in0502_6"	"An Emerging Consensus: Younger And Cohen Revisited"	"Younger, Barbara A. [meta:ra/0621053389 15]; Hollich, George [meta:ra/0621053389 16]; Furrer, Stephanie D. [meta:ra/0621053389 17]"	"2"	"5.0"	"issn:1525-0008 issn:1532-7078"	"209-216"	"2004-03-01"	"journal article"	"Wiley [meta:ra/0610116001 crossref:311]"	""	"Psychology"

structure of ERIH_META

Info about ERIH_META:
- 7622 files with self._interval = 10000;
- 32.69 GB

Answering to the research questions

4 Our research questions:
1. How many citations (according to **COCI**) involve, either as citing or cited entities, publications in SSH journals (according to **ERIH-PLUS**) included in OpenCitations **META**?
2. What are the disciplines that cites the most and those cited the most?
3. How many citations start from and go to publications in OpenCitations **META** that are not included in SSH journals?

4.1 ***Counter Class***
We have created a Python class, **Counter**, to answer to the three research questions. This class is able to execute two different methodologies, one that entails the production of output files (in this protocol it is referred to as "**Methodology1**"), reusable for other researches on the topic, and the other one that gives directly the answers to the questions ("**Methodology2**").
The constructor of the class requires three parameters:
- coci_preprocessed_path: Path to the directory that contains preprocessed COCI data (produced by CociPreProcessing, described in section 2.2).
- erih_meta_path: Path to the directory containing ERIH_META data. (produced by ErihMeta, described in section 3.1).
- num_cpus: number of cpu available for the execution of the program, by default it is set as the entire number of cpu available in the machine. This is also useful to define the number of threads to use for the execution of the program, which is defined as num_cpu * 4.

The methods of the class are:
- **get_all_files**: this method is derived from the one described in section 2 and used in the **Preprocessing** class. It retrieves all the files in the specified directory (***i_dir_or_compr***) that match the specified file extension (***req_type***). It returns a list of file paths.
- **splitted_to_file**: this is the same method described in section 2 and used in MetaPreProcessing and CociPreProcessing classes. It takes as input four parameters: ***cur_n***, the current count value; ***lines***, list of lines to write to a file; ***columns_to_use***, list of column names to use in the output file; ***output_dir_path***, output directory path. Writes the lines to a CSV file if the current count value is a multiple of a specified interval. Returns the remaining lines if not.
- **create_additional_files**: this method is executed if the user decides to use the methodology with the creation of output files. It infact

creates two datasets starting from ERIH_META: **erih_meta_with_disciplines** and **erih_meta_without_disciplines**, that will be described in the following sections.

- **create_disciplines_map**: this method is executed for answering to the second research question if the user decides to use the methodology with the creation of output files. It creates output files giving information about publications part of SSH journals, specifying the disciplines associated to them and a boolean value stating if they cite or are cited.
- **create_datasets_for_count**: this method is executed for answering to the first and third research question if the user decides to use the methodology with the creation of output files. In particular it creates two different datasets, **dataset_SSH** and **dataset_no_SSH**, that will be described in details later.
- **create_count_dictionaries**: this method is executed for answering to the second research question if the user decides to use the methodology with the creation of output files. It creates two dictionaries based on the output files of create_disciplines_map and counts the occurrences of disciplines in the citing and cited data.
- **count_lines**: this is executed for answering to the firts and third research question if the user decides to use the methodology with the creation of output files. It simply counts the total number of lines (excluding the columns' names) in the CSV files present in the specified directory.
- **iterate_erih_meta**: this method is executed if the user decides to use the faster methodology (the one that does not entail the production of files). It performs iteration over the ERIH_META data, building lists of DOIs for SSH (Social Sciences and Humanities) and non-SSH papers, as well as a mapping of ID to disciplines and a set of SSH disciplines.
- **execute_count**: this is the method that orchestrates the entire functioning of the class. It takes as input six parameters. The first is the path of the output folder where all the produced files will be stored (***output_dir***). The choice of the methodolgy to follow can be made through the second parameter (***create_subfiles***): if it is set to "True" a series of files, that are produced by the methods "**create_additional_files**" (responsible for creating "erih_meta_with_disciplines" and "erih_meta_without_disciplines") and "**create_datasets_for_count**" (it creates "dataset_SSH" and "dataset_no_SSH"), will be saved in subfolders inside the output folder specified by the user; if it is set to "False" the answers will be provided without producing any file. Then there are three boolean parameters (***answer_to_q1***, ***answer_to_q2***, ***answer_to_q3***) for choosing to which of the questions answe. The last parameter (***interval***) has the function to control the number of lines that will compose each file produced.

4.2 ***Methodology1 - create additional files*** *(answer to the first, second and third questions)*

The purpose of this method is to create two different subsets of ERIH_META data based on a given input parameter with_disciplines. These two subsets are then used for answering to the three research questions.
It determines the output directory (process_output_dir) and the columns to use (entity_columns_to_use) based on the with_disciplines parameter.
It initializes an empty list called data and a counter variable count to keep track of the number of records processed.

Erih_meta_with_disciplines

- If with_disciplines is True, it enters the first loop and iterates over the list containing **ERIH_META** files (self._list_erih_meta_files). It reads each file in chunks of 10,000 rows using pd.read_csv and selects the columns 'id' and 'erih_disciplines'. It fills any missing values in the chunk with an empty string. It converts the chunk into a list of dictionaries (df_dict_list). It iterates over each dictionary in the list and checks if the 'erih_disciplines' value is not empty. If the 'erih_disciplines' value is not empty, it splits the 'id' value (assuming it contains multiple IDs separated by spaces) and creates a new dictionary for each ID with the 'id' and 'erih_disciplines' values. The new dictionaries are appended to the data list, and the count is incremented. After processing a certain number of records (specified by self._interval), the data list is written to a file using the **splitted_to_file** method.

Erih_meta_without_disciplines

- If with_disciplines is False, it enters the second loop and follows a similar process as above, but this time it checks if the 'erih_disciplines' value is empty. If the 'erih_disciplines' value is empty, it splits the 'id' value and creates new dictionaries without the 'erih_disciplines' value. The new dictionaries are appended to the data list, and the count is incremented. After processing a certain number of records, the data list is written to a file using the **splitted_to_file** method.

Finally, if there are remaining records in the data list, it calculates the count needed to reach the next multiple of self._interval, and the remaining data is written to a file.

"id"	"erih_disciplines"
"doi:10.12759/hsr.46.2021.1.181-205"	"History, Interdisciplinary research in the Humanities, Interdisciplinary research in the Social Sciences, Sociology"

sample of erih_meta_with_disciplines subset

"id"
"doi:10.4230/lipics.approx/random.2020.19"

sample of erih_meta_without_disciplines subset

4.3 ***Methodology1 - create_datasets_for_count*** *(answer to the first and third questions)*

This method creates two different datasets (**dataset_SSH**, **dataset_no_SSH**) based on the processing of **COCI_preprocessed**, **erih_meta_with_disciplines** and **erih_meta_without_disciplines**. These datasets are used to answer the first and third research questions. Each dataset has four columns: "citing", "is_citing_SSH", "cited", and "is_cited_SSH". The "is_citing_SSH" and "is_cited_SSH" columns contain boolean values: "True" if the corresponding entity is associated with a SSH (Social Sciences and Humanities) discipline and "False" otherwise.

The method first of all determines the output directory (output_process_dir) based on the is_SSH parameter.

- If is_SSH is True, it initializes an instance of the CSVManager class with the path to the "erih_meta_with_disciplines" file (self._path_erih_meta_with_disciplines).
- If is_SSH is False, it loads the "erih_meta_without_disciplines" file as a set of IDs (self._set_erih_meta_without_disciplines).

It initializes an empty list called data and a counter variable count to keep track of the number of records processed. It iterates over a list of files (self._list_coci_files). It reads each file in chunks of 10,000 rows using pd.read_csv. It fills any missing values in the chunk with an empty string.

It converts the chunk into a list of dictionaries (df_dict_list).

It iterates over each dictionary in the list and retrieves the values for "citing" and "cited".

Depending on the value of is_SSH, it checks if either the citing or cited entity satisfies certain conditions.

- If is_SSH is True, it checks if either the citing or cited entity is present in the "erih_meta_with_disciplines" dataset.
- If is_SSH is False, it checks if both the citing and cited entities are present in the "erih_meta_without_disciplines" set.

If the condition is met, the counter count is incremented.

Depending on the values of is_SSH and the entity conditions, dictionaries are created and appended to the data list.

After processing a certain number of records (specified by self._interval), the data list is written to a file using the **splitted_to_file** method.

If there are remaining records in the data list, it calculates the count needed to reach the next multiple of self._interval, and the remaining data is written to a file.

"citing"	"is_citing_SSH"	"cited"	"is_cited_SSH"
"doi:10.1002/9781118541203.xen0011"	"False"	"doi:10.1073/pnas.93.10.4644"	"True"

structure of the two datasets dataset_SSH and dataset_no_SSH

4.4 ***Methodology 1 - create_disciplines_map*** *(answer to the second question)*

This method creates output CSV files, starting from **COCI_preprocessed** and **erih_meta_with_disciplines**, with four columns ("id", "citing", "cited", "disciplines") giving information about publications part of SSH journals, specifying the disciplines associated to them and a boolean value stating if they cite or are cited.

It initializes an instance of the **CSVManager** class with the path to the "erih_meta_with_disciplines" file (self._path_erih_meta_with_disciplines).

It initializes an empty list called data and a counter variable count to keep track of the number of records processed. It iterates over a list containing **COCI_preprocessed** files (self._list_coci_files).

It reads each file in chunks of 10,000 rows using pd.read_csv. It fills any missing values in the chunk with an empty string. It converts the chunk into a list of dictionaries (df_dict_list).

It iterates over each dictionary in the list and retrieves the values for "citing" and "cited".

It checks if either the citing or cited entity is present in the "erih_meta_with_disciplines" dataset using the **get_value** method of the CSVManager class.

If the condition is met, it retrieves the associated disciplines for the citing or cited entity.

It splits the string of disciplines into a list and iterates over each discipline.

For each discipline, it creates a dictionary containing the ID, the role (citing or cited), and the discipline itself.

The dictionary is appended to the data list, and the counter count is incremented.

After processing a certain number of records (specified by self._interval), the data list is written to a file using the **splitted_to_file** method.

If there are remaining records in the data list, it calculates the count needed to reach the next multiple of self._interval, and the remaining data is written to a file.

4.5 ***Methodology 1 - create_count_dictionaries (answer to the second question)***

The method uses the data produced by **create_disciplines_map** (self._path_dataset_map_disciplines). It counts starting from two dictionaries the occurrences of disciplines for citing and cited entities, and returns the maximum count values and associated disciplines for both types of entities.

It initializes two empty dictionaries: dict_citing and dict_cited.

It iterates over the files in the directory specified by self._path_dataset_map_disciplines, filtering for files with the .csv extension. For each

file, it reads the data in chunks of 10,000 rows using pd.read_csv. It fills any missing values in the chunk with an empty string. It converts the chunk into a list of dictionaries (df_dict_list).
It iterates over each dictionary in the list and checks if it represents a citing or cited entity.

- If it represents a citing entity (based on the value of the 'citing' key), it retrieves the associated discipline from the 'disciplines' key. If the discipline is not already a key in the dict_citing dictionary, it adds the discipline as a key with a value of 1. If the discipline is already a key, it increments the corresponding value by 1.
- If it represents a cited entity (based on the value of the 'cited' key), it performs the same process as above for the dict_cited dictionary.

After processing all the dictionaries in the chunks from each file, it determines the maximum count value in both dict_citing and dict_cited using the max function on the dictionary values.
It finds the discipline associated with the maximum count in both dict_citing and dict_cited using the max function on the dictionary keys, with the get method as the key argument.
It returns four values: the maximum count for citing entities (max_value_citing), the maximum count for cited entities (max_value_cited), the discipline associated with the maximum count for cited entities (max_discipline_cited), and the discipline associated with the maximum count for citing entities (max_discipline_citing).
It returns also the two dictionaries obtained: citing_dict and cited_dict.

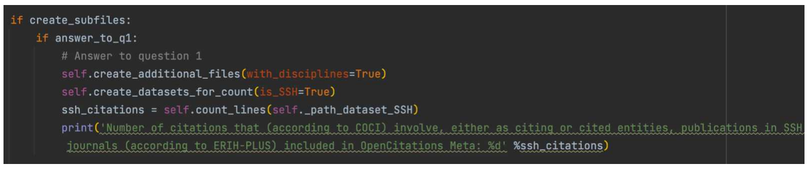
4.6 ***Methodology2 - iterate_erih_meta***
This method processes **ERIH_META** data, extracts DOIs and their associated disciplines, creates sets of unique DOIs for SSH and non-SSH papers, and returns these sets along with a dictionary mapping DOIs to their disciplines and a set of unique SSH disciplines.
It initializes several variables: ssh_papers (an empty list to store DOIs associated with SSH disciplines), not_ssh_papers (an empty list to store DOIs not associated with SSH disciplines), id_disciplines_map (an empty dictionary to store DOIs as keys and their associated disciplines as values), and ssh_disciplines (an empty set to store unique SSH disciplines).
It iterates over the filenames in self._list_erih_meta_files.
For each filename, it reads a CSV file into a DataFrame (df), selecting only the 'id' and 'erih_disciplines' columns.
It fills any NaN or None values in the DataFrame with an empty string.
It creates a boolean mask (mask) based on the non-empty values in the 'erih_disciplines' column.
It filters the DataFrame (ssh_df) using the mask, representing rows with SSH disciplines.
It resets the index of ssh_df for iteration.
It iterates over each row in ssh_df.
It splits the disciplines into a list by comma separation and removes leading/trailing whitespace.
It retrieves the DOI from the 'id' column.
If the DOI is not in id_disciplines_map, it adds it as a key with the disciplines as the value list.
Otherwise, it extends the existing list of disciplines for the DOI.
It checks each discipline and adds it to the ssh_disciplines set if it is not already present.
It creates another DataFrame (not_ssh_df) using the inverse of the mask, representing rows without SSH disciplines.
It resets the index of not_ssh_df for iteration.
It retrieves the unique values of the 'id' column from ssh_df and not_ssh_df, storing them in unique_ssh and unique_not_ssh lists, respectively.
It extends the ssh_papers list with the unique SSH DOIs and extends the not_ssh_papers list with the unique non-SSH DOIs.
It splits each DOI in ssh_papers and not_ssh_papers by whitespace and extends the respective unique lists (ssh_papers_unique and not_ssh_papers_unique).
It creates a new dictionary (unique_id_disciplines_map) by splitting the keys in id_disciplines_map and associating each split key with its value.
It creates sets (ssh_set and not_ssh_set) from ssh_papers_unique and not_ssh_papers_unique, respectively.
It returns four values: ssh_set (set of unique SSH DOIs), not_ssh_set (set of unique non-SSH DOIs), unique_id_disciplines_map (dictionary mapping unique DOIs to their associated disciplines), and ssh_disciplines (set of unique SSH disciplines).

4.7

Execute_count
As said before, this is the main method of the class **Counter**, the one that orchestrates the entire process by calling the methods already described in details.
The execution of the first or of the second methodology is controlled by the parameter "create_subfiles" and it is managed through an if-else statement (if create_subfiles -> Methodlogy1, else -> Methodology2)

Methodology1

First question
It calls the method **create_additional_files** with the parameter with_disciplines set to True, to produce the **erih_meta_with_disciplines** subset.
Then the **dataset_SSH** is created through the method **create_datasets_for_count**, with the parameter is_SSH set to True.
To retrieve the number of citations that involve publications in SSH journals the lines of the files part of the dataset_SSH are counted, using the method **count_lines**.

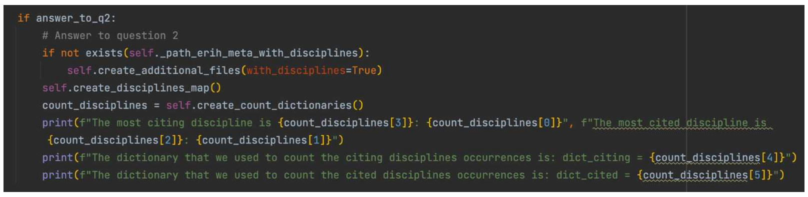


```
if create_subfiles:
    if answer_to_q1:
        # Answer to question 1
        self.create_additional_files(with_disciplines=True)
        self.create_datasets_for_count(is_SSH=True)
        ssh_citations = self.count_lines(self._path_dataset_SSH)
        print('Number of citations that (according to COCI) involve, either as citing or cited entities, publications in SSH
         journals (according to ERIH-PLUS) included in OpenCitations Meta: %d' %ssh_citations)
```

Answer to the first question

Second question

First of all it checks it the subset **erih_meta_with_disciplines** has already been created, if not it calls the method **create_additional_files** with the parameter with_disciplines set to True, to produce the erih_meta_with_disciplines subset. Then the methods **create_disciplines_map** and **create_count_dictionaries** are executed. The four outputs plus the dictionaries of create_count_dictionaries are stored in a variable (count_disciplines).

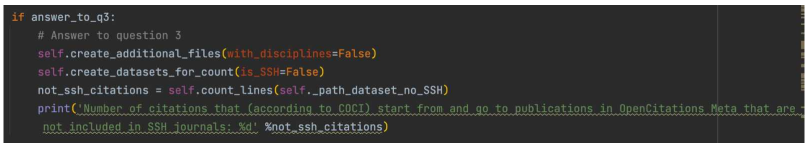


```
if answer_to_q2:
    # Answer to question 2
    if not exists(self._path_erih_meta_with_disciplines):
        self.create_additional_files(with_disciplines=True)
    self.create_disciplines_map()
    count_disciplines = self.create_count_dictionaries()
    print(f"The most citing discipline is {count_disciplines[3]}: {count_disciplines[0]}", f"The most cited discipline is
     {count_disciplines[2]}: {count_disciplines[1]}")
    print(f"The dictionary that we used to count the citing disciplines occurrences is: dict_citing = {count_disciplines[4]}")
    print(f"The dictionary that we used to count the cited disciplines occurrences is: dict_cited = {count_disciplines[5]}")
```

Answer to question 2

Third question

It calls the method **create_additional_files** with the parameter with_disciplines set to False, to produce the **erih_meta_without_disciplines** subset. Then the **dataset_no_SSH** is created through the method **create_datasets_for_count**, with the parameter is_SSH set to False.
To retrieve the number of citations that involve publications not included in SSH journals the lines of the files part of the dataset_no_SSH are counted, using the method **count_lines**.



```
if answer_to_q3:
    # Answer to question 3
    self.create_additional_files(with_disciplines=False)
    self.create_datasets_for_count(is_SSH=False)
    not_ssh_citations = self.count_lines(self._path_dataset_no_SSH)
    print('Number of citations that (according to COCI) start from and go to publications in OpenCitations Meta that are
     not included in SSH journals: %d' %not_ssh_citations)
```

Answer to the third question

Methodology2

It initializes the ssh_set, not_ssh_set, id_disciplines_map, and ssh_disciplines variables by calling the **iterate_erih_meta** method.
It initializes the variables ssh_citations and not_ssh_citations to track the count of SSH and non-SSH citations, respectively.
It defines a nested function **count_citations_in_file** that takes in id_disciplines_map, ssh_disciplines, ssh_set, not_ssh_set, and a file path as input. This function reads a CSV file (from COCI) specified by the file path, retrieves the 'citing' and 'cited' columns, and iterates over the data to count the citations based on the citation type and discipline. It returns a dictionary discipline_counter that counts the occurrences of disciplines, the count of SSH citations, and the count of non-SSH citations.
Then it starts the counting process. Using a ThreadPoolExecutor with a maximum number of workers specified by self.num_threads, it applies the count_citations_in_file function to each file in COCI_preprocessed (self._list_coci_files), passing the required arguments and storing the results in a list called results.
It initializes the discipline_counter dictionary and updates the ssh_citations and not_ssh_citations counts by iterating over the results.
If all three answers to the questions are requested (answer_to_q1, answer_to_q2, and answer_to_q3 are True), it calculates the discipline with the highest count of citing and cited papers (discipline_with_highest_citing and discipline_with_highest_cited, respectively).
If only specific answers are requested, it calculates the relevant results based on the conditions and prints them accordingly.
It returns the counts of SSH and non-SSH citations (ssh_citations and not_ssh_citations), the discipline that cites the most (discipline_with_highest_citing), the most cited discipline (discipline_with_highest_cited), and the discipline_counter dictionary that tracks the counts of disciplines citations, both done and received.

Results

5 The answers to the three questions gives significant insights on the current situations within publications in SSH Journals.

All the graphical visualizations have been realised with Plotly (https://chart-studio.plotly.com/feed/#/).

5.1 ***The answer to the first question***

How many citations (according to COCI) involve, either as citing or cited entities, publications in SSH journals (according to ERIH-PLUS) included in OpenCitations META?

For answering to this question, using the Methodology1, we have produced erih_meta_with_disciplines (450 MB unzipped; 550 files with self._interval = 10000) and dataset_SSH (17.54 GB unzipped; 22030 files with self._interval = 10000).

The citations which involve publications in SSH journals according to the three datasets are 220295011.

We have realised a graphical visualization for this answer, together with the third one, which can be found here: https://opensciencepika.github.io/results/Plot4.html

5.2 ***The answer to the second question***

What are the disciplines that cites the most and those cited the most?

For answering to this question, using the Methodology1, we use erih_meta_with_disciplines and we produce dataset_map_disciplines (46.32 GB unzipped; 67380 files with self._interval = 10000)

The discipline citing the most is Psychology with 54512160 citations, while the discipline cited the most is still Psychology with 83291583.

We have realised a graphical visualization for this answer, which can be found here: https://opensciencepika.github.io/results/index.html#first (for citing disciplines), https://opensciencepika.github.io/results/Plot2.html (for cited disciplines).

5.3 ***The answer to the third question***

How many citations start from and go to publications in OpenCitations **META** that are not included in SSH journals?

For answering to this question, using the Methodology1, we create erih_meta_without_disciplines (2.34 GB unzipped; 7073 files with self._interval = 10000) and dataset_no_SSH (95.57 GB unzipped; 117639 files with self._interval = 10000)

The number of citations that (according to COCI) start from and go to publications in OpenCitations Meta that are not included in SSH journals is 1176384557.

We have realised a graphical visualization for this answer, together with the first one, which can be found here: https://opensciencepika.github.io/results/Plot4.html