Uncovering the Citation Landscape: Exploring COCI, Meta, and ERIH-PLUS in Social Sciences and Humanities Journals

Lorenzo Paolini

Data Curation - Scrub data for initial use
Methodology - Development and design methodology
Software - Software development
Validation - Verification of reproducibility of results

Olga Pagnotta

Data Curation - Scrub data for initial use
Methodology - Development and design methodology
Software - Software development
Validation - Verification of reproducibility of results
Visualization - Graphical visualization of the results
Writing - Review & Editing

Marta Soricetti

Data Curation - Scrub data for initial use
Methodology - Development and design methodology
Software - Software development
Visualization - Graphical visualization of the results
Writing - Review & Editing

Sara Vellone

Data Curation - Scrub data for initial use
Investigation - Research and investigation of previous literature
Methodology - Development and design methodology
Writing - Original Draft
Writing - Review & Editing

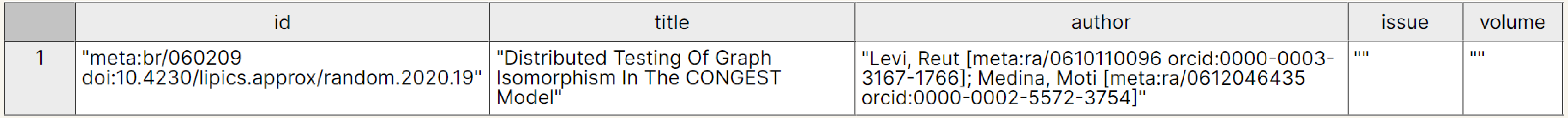
Open Science's Final Project

Alma Mater Studiorum - Digital Humanities and Digital Knowledge

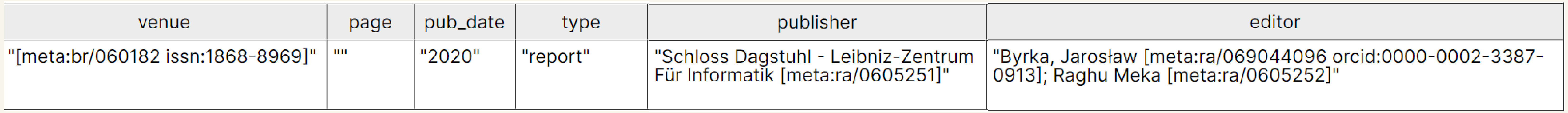
A.Y. 2022/2023

OPEN CITATION META

OpenCitations Meta stores and delivers bibliographic metadata for all publications involved in the OpenCitations Indexes.



	id	title	author	issue	volume
1	"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]"	""	""

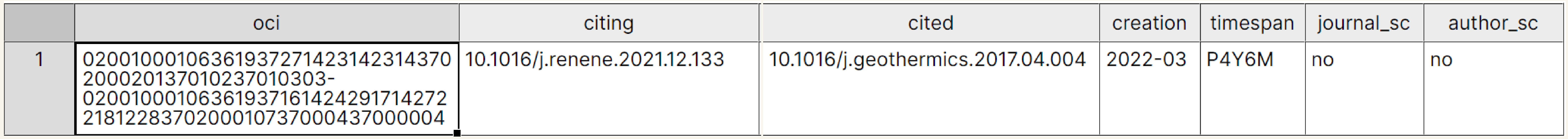


venue	page	pub_date	type	publisher	editor
"[meta:br/060182 issn:1868-8969]"	""	"2020"	"report"	"Schloss Dagstuhl - Leibniz-Zentrum Für Informatik [meta:ra/0605251]"	"Byrka, Jarosław [meta:ra/069044096 orcid:0000-0002-3387-0913]; Raghu Meka [meta:ra/0605252]"

THE DATASETS

OPEN CITATION COCI

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.

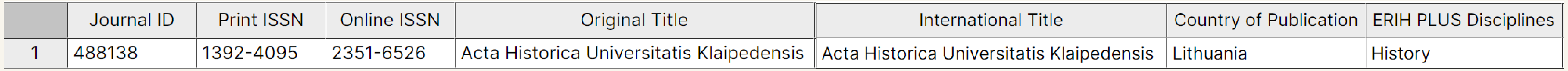


	oci	citing	cited	creation	timespan	journal_sc	author_sc
1	02001000106361937271423142314370200020137010237010303-02001000106361937161424291714272218122837020001073700043700000 4	10.1016/j.renene.2021.12.133	10.1016/j.geothermics.2017.04.004	2022-03	P4Y6M	no	no

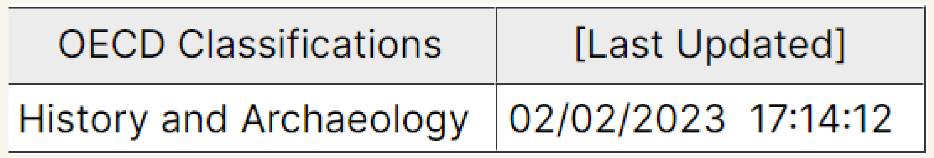
THE DATASETS

ERIH-PLUS

ERIH PLUS is an academic journal index for the SSH (Social Sciences and Humanities) society in Europe.



	Journal ID	Print ISSN	Online ISSN	Original Title	International Title	Country of Publication	ERIH PLUS Disciplines
1	488138	1392-4095	2351-6526	Acta Historica Universitatis Klaipedensis	Acta Historica Universitatis Klaipedensis	Lithuania	History



OECD Classifications	[Last Updated]
History and Archaeology	02/02/2023 17:14:12

THE DATASETS

THE RESEARCH QUESTIONS

Q1: 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;

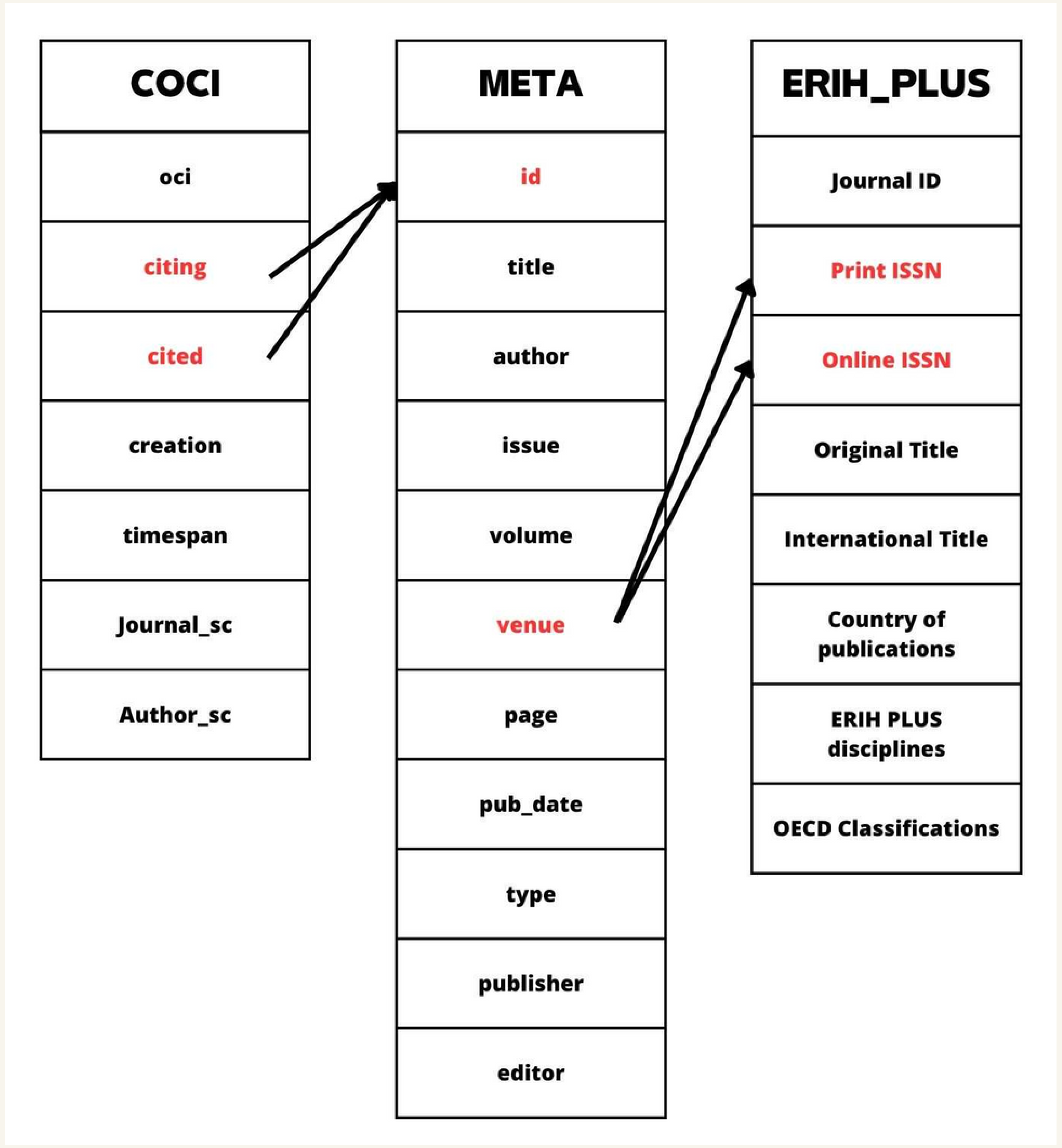
Q2: The most citing and the most cited SSH discipline, according to the above mentioned datasets;

Q3: The citations coming from and going to publications contained in OpenCitations Meta which are not included in SSH journals.

THE SOFTWARE

-

PREPROCESS



As a first step, we have tried to find a matching between the different datasets, thus discovering the data they have in common.

Meta represents the joining link between COCI and ERIH PLUS, allowing to retrieve information about the publication discipline starting from a DOI (classified as citing or cited).

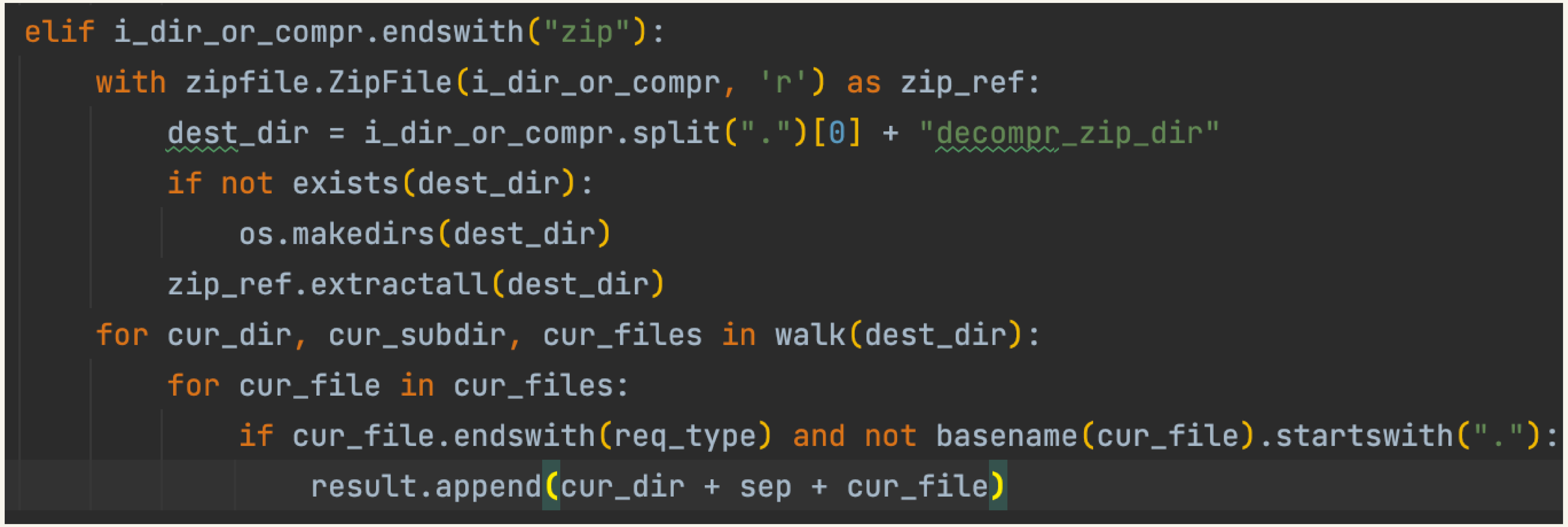
THE SOFTWARE

-

PREPROCESS

Preprocessing (superclass)

For the preprocessing of META and COCI datasets we have created a Superclass, called **Preprocessing**, which contains the method "**get_all_files**", that is responsible of retrieving all the files from the original dump by decompressing, extracting, and iterating over nested directories. This method takes in input a path (i_dir_or_compr), the string of a file extension (req_type) and returns a list of all the files with the required extension found in the data dump (either a directory or a compressed archive), and a "tar.gz" opener if needed, otherwise None (if the dump is not tar.gz compressed it is not needed).



```
elif i_dir_or_compr.endswith("zip"):
    with zipfile.ZipFile(i_dir_or_compr, 'r') as zip_ref:
        dest_dir = i_dir_or_compr.split(".")[0] + "decompr_zip_dir"
        if not exists(dest_dir):
            os.makedirs(dest_dir)
        zip_ref.extractall(dest_dir)
    for cur_dir, cur_subdir, cur_files in walk(dest_dir):
        for cur_file in cur_files:
            if cur_file.endswith(req_type) and not basename(cur_file).startswith("."):
                result.append(cur_dir + sep + cur_file)
```

https://archive.softwareheritage.org/swh:1:cnt:2faf157225885e5420cdd740bee5311649c1b1a1;origin=https://pypi.org/project/oc-preprocessing/;visit=swh:1:snp:b429746305d915b577b0ed022b2650b70ecf5dc2;anchor=swh:1:rel:44fb3b0a058877ea4ef15490a499391c910a384f;path=/oc_preprocessing-0.0.5/preprocessing/base.py;lines=14

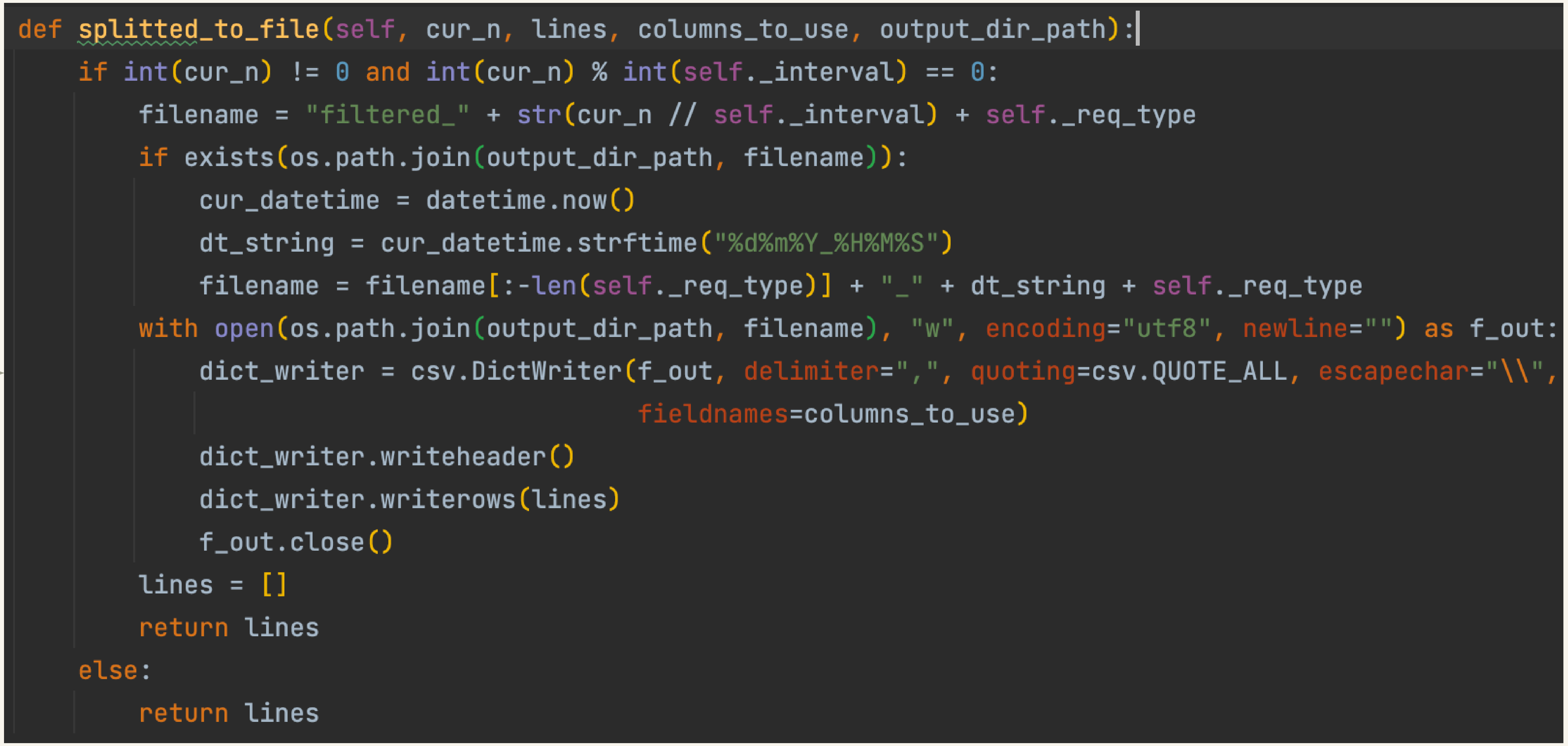
THE SOFTWARE

-

PREPROCESS

MetaPreProcessing and CociPreProcessing

MetaPreProcessing and **CociPreProcessing** inherit from the Preprocessing class. These classes share the method **splitted_to_file**, responsible for concretely producing the output files with the processed data obtained with 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.



```python
def splitted_to_file(self, cur_n, lines, columns_to_use, output_dir_path):
    if int(cur_n) != 0 and int(cur_n) % int(self._interval) == 0:
        filename = "filtered_" + str(cur_n // self._interval) + self._req_type
        if exists(os.path.join(output_dir_path, filename)):
            cur_datetime = datetime.now()
            dt_string = cur_datetime.strftime("%d%m%Y_%H%M%S")
            filename = filename[:-len(self._req_type)] + "_" + dt_string + self._req_type
        with open(os.path.join(output_dir_path, filename), "w", encoding="utf8", newline="") as f_out:
            dict_writer = csv.DictWriter(f_out, delimiter=",", quoting=csv.QUOTE_ALL, escapechar="\\",
                                         fieldnames=columns_to_use)
            dict_writer.writeheader()
            dict_writer.writerows(lines)
            f_out.close()
        lines = []
        return lines
    else:
        return lines
```

"https://archive.softwareheritage.org/swh:1:cnt:2faf157225885e5420cdd740bee5311649c1b1a1;origin=https://pypi.org/project/oc-preprocessing/;visit=swh:1:snp:b429746305d915b577b0ed022b2650b70ecf5dc2;anchor=swh:1:rel:44fb3b0a058877ea4ef15490a499391c910a384f;path=/oc_preprocessing-0.0.5/preprocessing/base.py;lines=141

THE SOFTWARE

-

PREPROCESS

MetaPreProcessing

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.
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**). These CSVs are then used in the **CociPreProcessing** class.

"id": before processing	**"id": after processing**
"meta:br/060209 doi:10.4230/lipics.approx/random.2020.19"	"doi:10.4230/lipics.approx/random.2020.19"

THE SOFTWARE

-

PREPROCESS

CociPreProcessing

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 outupt 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**.

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.

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

THE SOFTWARE - PREPROCESS

ErihPreProcessing

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.

Print ISSN	**Online ISSN**	**after: venue_id**
1392-4095	2351-6526	"issn:1392-4095 issn:2351-6526"

THE SOFTWARE

-

ERIH-META

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, using the class **ErihMeta**, obtaining new CSVs having as columns all the columns of META with the addition of the ERIH-PLUS disciplines.

To retrieve the disciplines associated to a specific venue id we have used the class **CSVManager** of OpenCitations, and in particular the method **get_value**, adapting the whole class to the structure of the output file of ErihPreProcessing.

https://archive.softwareheritage.org/swh:1:cnt:8362d20804ab87cf6862ffe37f2624e26634eff2;origin=https://github.com/opencitations/oc_meta;visit=swh:1:snp:dcbd1d5baf07b98504292230c53215f1e080e782;anchor=swh:1:rev:8510cdd12e3693202231fef27b0311ca7b88414d;path=/oc_meta/lib/csvmanager.py;lines=24

THE SOFTWARE
-
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 ("**Methodology1**"), reusable for other researches on the topic, and the other one that gives directly the answers to the questions ("**Methodology2**").
The class uses two already mentioned methods: **get_all_files** (a smaller version) and **splitted_to_file.**

The main method of the class is **execute_count.** It orchestrates the entire functioning of the class. It takes as input **six parameters**.

- **output_dir**: the path of the output folder where all the produced files will be stored
- **create_subfiles**: if it is set to "True" a series of files, produced by the methods "create_additional_files", "create_datasets_for_count" and "create_disciplines_map" , will be saved in subfolders inside the output folder specified by the user (**Methodology1**); if it is set to "False" the answers will be provided without producing any file (**Methodology2**)
- **answer_to_q1, answer_to_q2, answer_to_q3:** boolean parameters to choose the answer to produce.
- **interval**: to control the number of lines that will compose each file produced.

THE SOFTWARE

-

COUNTER CLASS

Methodology 1

create additional files

It creates 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.

Erih_meta_with_disciplines

If with_disciplines is True, it iterates over **ERIH_META** and, for every "id" which has a value in the "erih_disciplines" column, it writes a new line in the output file, which will have two columns: the id and the corresponding discipline.

Erih_meta_without_disciplines

If with_disciplines is False, the files are produced by taking all the ids without a discipline associated to them. The output files have only the id column.

THE SOFTWARE

-

COUNTER CLASS

Methodology 1

Erih_meta_with_disciplines

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"

Erih_meta_without_disciplines

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

THE SOFTWARE

-

COUNTER CLASS

Methodology 1

Questions 1 & 3

create_datasets_for_count
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 discipline and "False" otherwise.

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

The answers to the questions is then given by counting all the lines of the output files with the method **count_lines**.

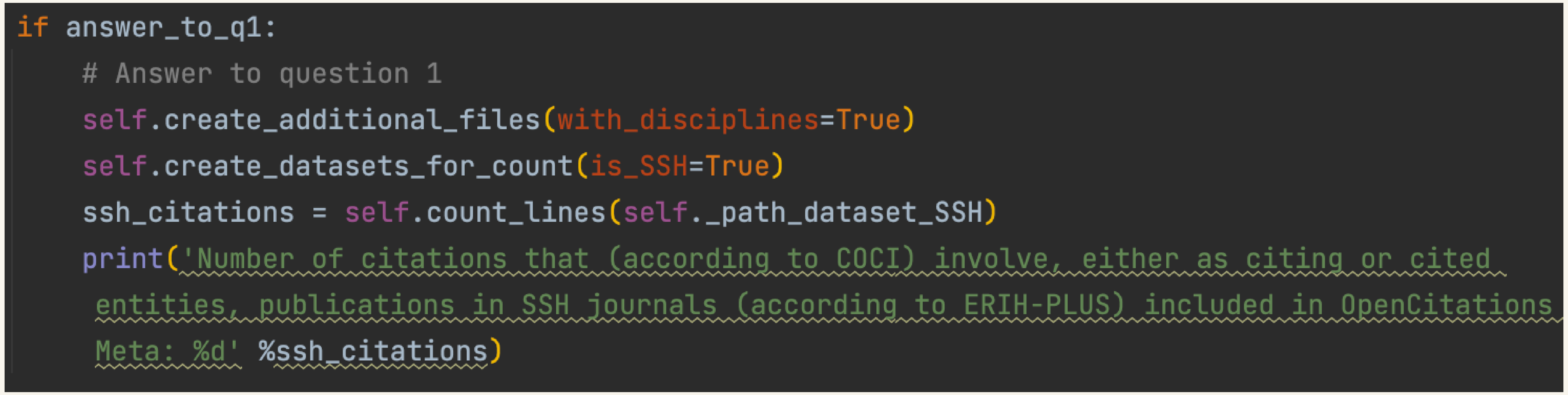
THE SOFTWARE

-

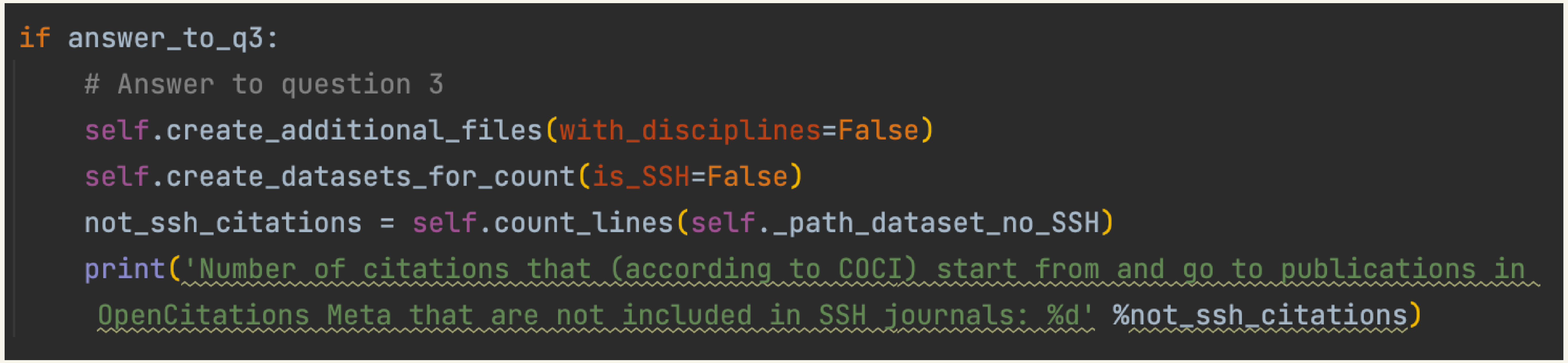
COUNTER CLASS

Methodology 1

Questions 1 & 3



```
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)
```



```
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)
```

THE SOFTWARE

-

COUNTER CLASS

Methodology 1

Question 2

create_disciplines_map
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.

id	citing	cited	disciplines
"doi:10.1073/pnas.93.10.4644"	"False"	"True"	"Anthropology"

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

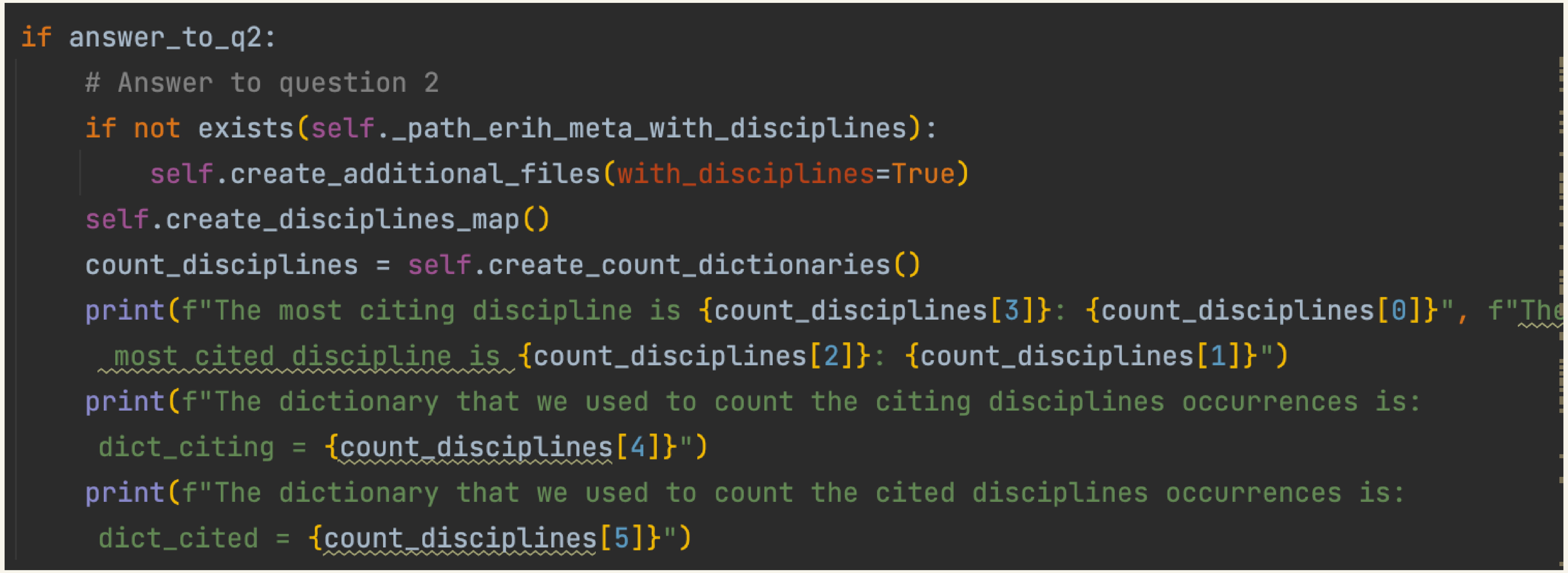
THE SOFTWARE

-

COUNTER CLASS

Methodology 1

Questions 2



```
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]}")
```

THE SOFTWARE

-

COUNTER CLASS

Methodology 2

A "lighter" method

By running the ***execute_count*** method with the default parameters (i.e. without specifying anything), you will get access to a faster execution of the counts. Such execution will directly produce and print on screen the answers to the three research questions, or to just a subset of them.

A quick step-by-step guide

This second option involves **3 main steps**:

- First, we iterate over the previously built Erih_Meta.csv files to extract the relevant information in a light and efficient way;
- Next, we use Multi Threading to execute the different operations necessary to obtain the answers;
- Finally, we print the results on the screen and return the main variables to the user.

Let's quickly unpack them...

THE SOFTWARE

-

COUNTER CLASS

Methodology 2

First step >>> Iterate over Erih_Meta

This first step involves reading all the files produced by the merged OC_Meta and Erih_Plus files. Each file will be firstly ***masked*** in order to build DataFrames containing either all the SSH-related papers, or all the others.

From each SSH-related paper, we ***map its DOI* with the disciplines** it is related with inside a dictionary that has DOIs as keys, and lists of single disciplines as values. Additionally, from the same DataFrame, we **extract all the *unique disciplines*** and store them into a set.

Additionally, a common operation that involves both the DataFrames is the **extraction of all the unique DOIs**, which are then stored inside two additional sets. At the end of the process, each of these sets will contain **unique SSH-related DOIs**, and **unique non-SSH-related DOIs**.

All these objects (except for the set containing disciplines) will be **checked against double identifiers**, which may refer to the same publication in different formats. In such cases the DOIs will be unpacked.

THE SOFTWARE

-

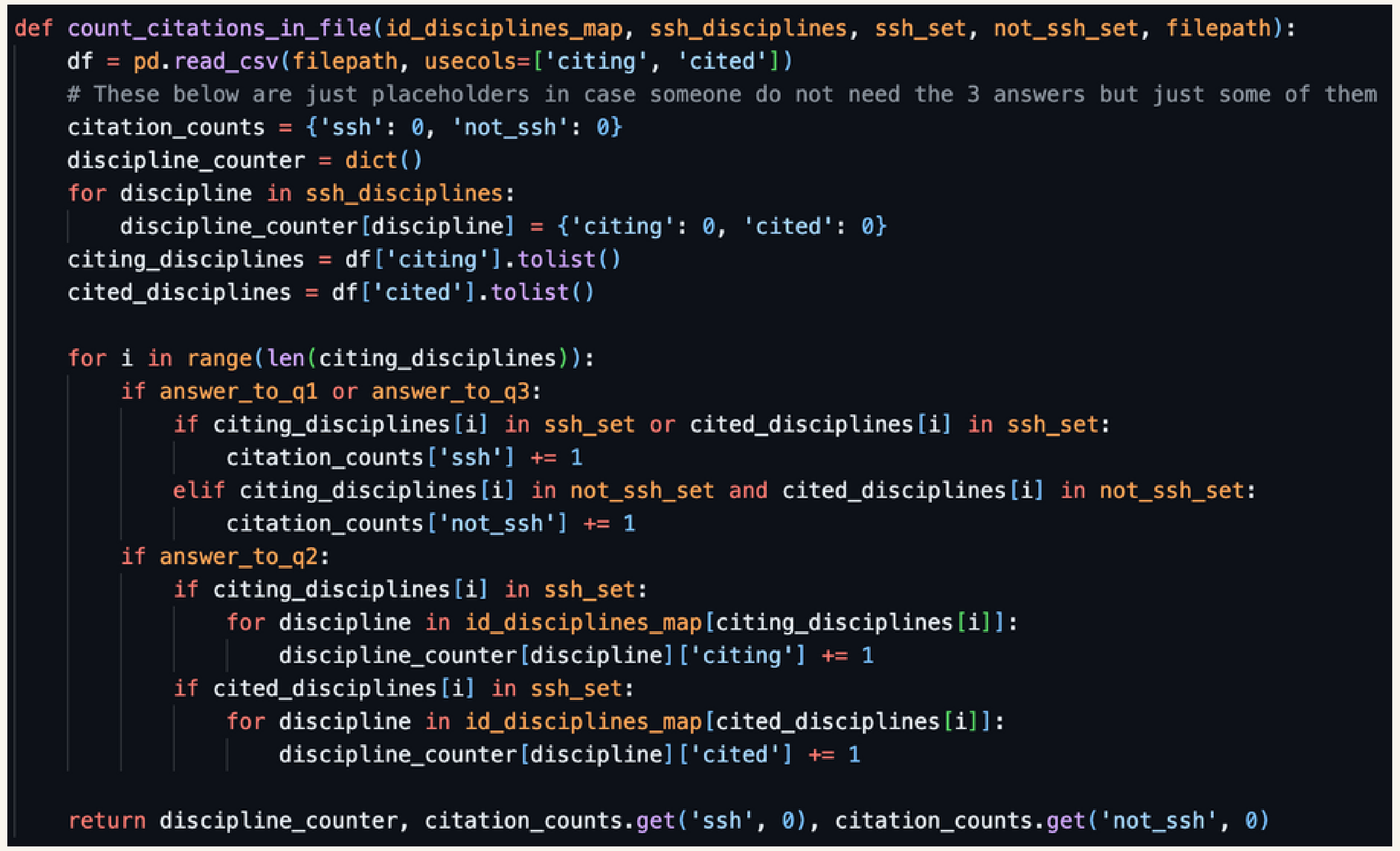
COUNTER CLASS

Methodology 2

Second step >>> Let's count

This second step is essentially implemented as a **single, straightforward function** that is invoked using a **multithreaded approach**.

Basically, the function **concurrently reads** (thanks to multithreading approach) **the files from the processed version of COCI**, turns each column into a list (this is done in order to speed up the following for-loop, which is over the length of these lists instead of being over the rows of DataFrames), and updates some counter objects. These objects are finally read by the main method, which extract the single results from each file, in order to update some **general-purpose counters**, which are the printed on screen and returned with the last step (**third step >>> print everything**).



```python
def count_citations_in_file(id_disciplines_map, ssh_disciplines, ssh_set, not_ssh_set, filepath):
    df = pd.read_csv(filepath, usecols=['citing', 'cited'])
    # These below are just placeholders in case someone do not need the 3 answers but just some of them
    citation_counts = {'ssh': 0, 'not_ssh': 0}
    discipline_counter = dict()
    for discipline in ssh_disciplines:
        discipline_counter[discipline] = {'citing': 0, 'cited': 0}
    citing_disciplines = df['citing'].tolist()
    cited_disciplines = df['cited'].tolist()

    for i in range(len(citing_disciplines)):
        if answer_to_q1 or answer_to_q3:
            if citing_disciplines[i] in ssh_set or cited_disciplines[i] in ssh_set:
                citation_counts['ssh'] += 1
            elif citing_disciplines[i] in not_ssh_set and cited_disciplines[i] in not_ssh_set:
                citation_counts['not_ssh'] += 1
        if answer_to_q2:
            if citing_disciplines[i] in ssh_set:
                for discipline in id_disciplines_map[citing_disciplines[i]]:
                    discipline_counter[discipline]['citing'] += 1
            if cited_disciplines[i] in ssh_set:
                for discipline in id_disciplines_map[cited_disciplines[i]]:
                    discipline_counter[discipline]['cited'] += 1

    return discipline_counter, citation_counts.get('ssh', 0), citation_counts.get('not_ssh', 0)
```

Each reading operation is shown on screen thanks to ***tqdm*** progress bars.

THE SOFTWARE

-

run.py

Automate the process

Together with the methods explained before, we also built a ***requirements.txt*** and a ***run.py*** file, useful to automate the entire workflow we have followed, and showed, so far.

The latter file, just needs to be launched, together with some user defined parameters (such as file paths, or processing options), and will take into account all the operations we have so far described.

Further improvements to this file are possible, such as running in parallel preprocessing operations (which could also be improved and singularly parallelized themself), or also add an automatic download of the files required to tale into account the whole process.

Some efforts have also been done in order to write the complete code automation in *bash*, but we didn't managed to finish the code for testing and time issues.

THE SOFTWARE

-

Hardware requirements

***Storage* :** The big amount of files requires an hardware capable of dealing with them. In particular, for what concerns the minimum storage to reproduce our experiment, we can estimate the need for **355 GB** of available space just for *raw and processed files* (without considering any deletion of used files). Additionally, if you want to produce also the *additional files* depicted in methodology1, you will need other **165 GB**, for an *overall amount* of around **520 GB**.

***RAM* :** According to our experiments, a minimum amount of **16 GB** is required.

***CPU* :** According to our experiments, we suggest at least **8 cores**.

***OS* :** Any.

***GPU* :** No formal requirements, it is not used.

ANSWER 1

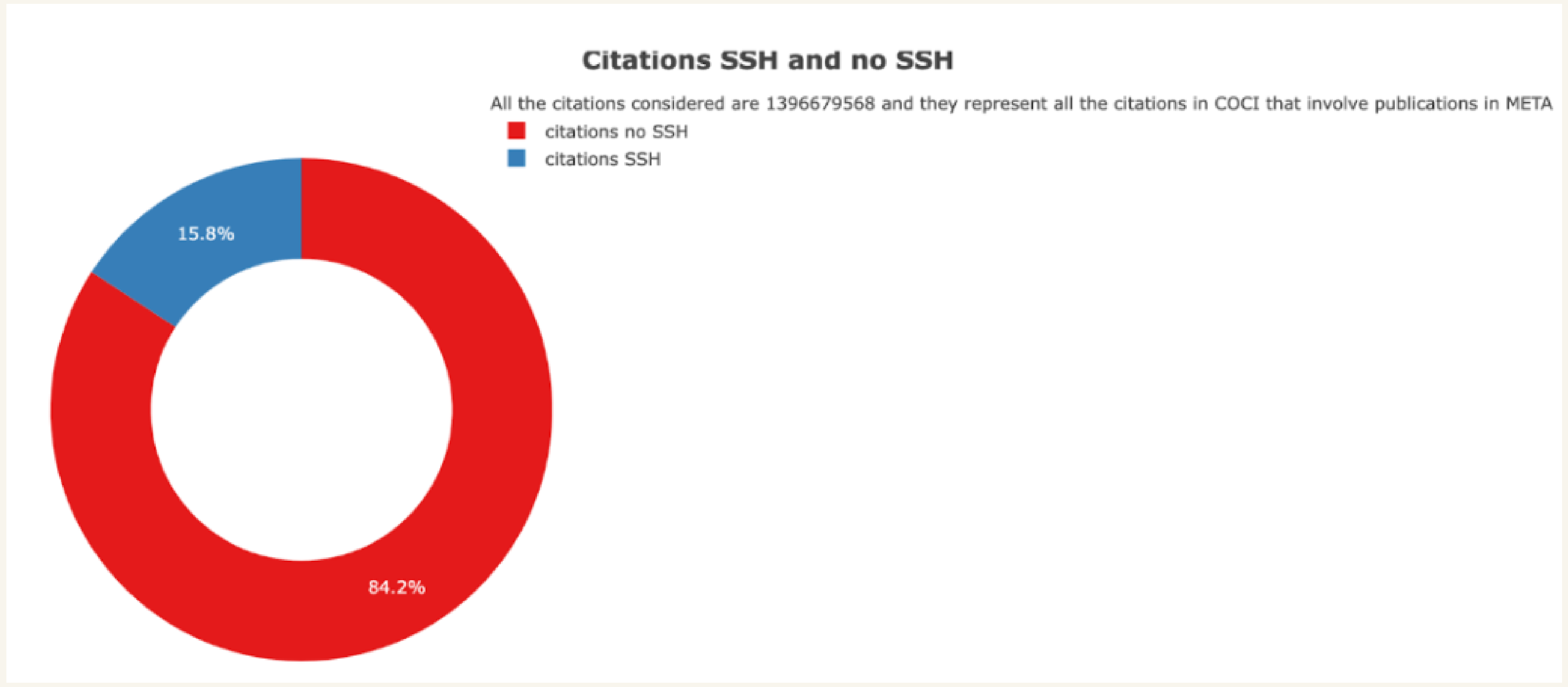
The number of citations referring to publication in SSH journals included in ERIH-PLUS, by looking at citations data contained in COCI and Meta is of 220.295.011 citations.

THE RESULTS

See our results here: https://opensciencepika.github.io/results/

ANSWER 3

The number of citations that are not included in SSH journals is 1.176.384.557.

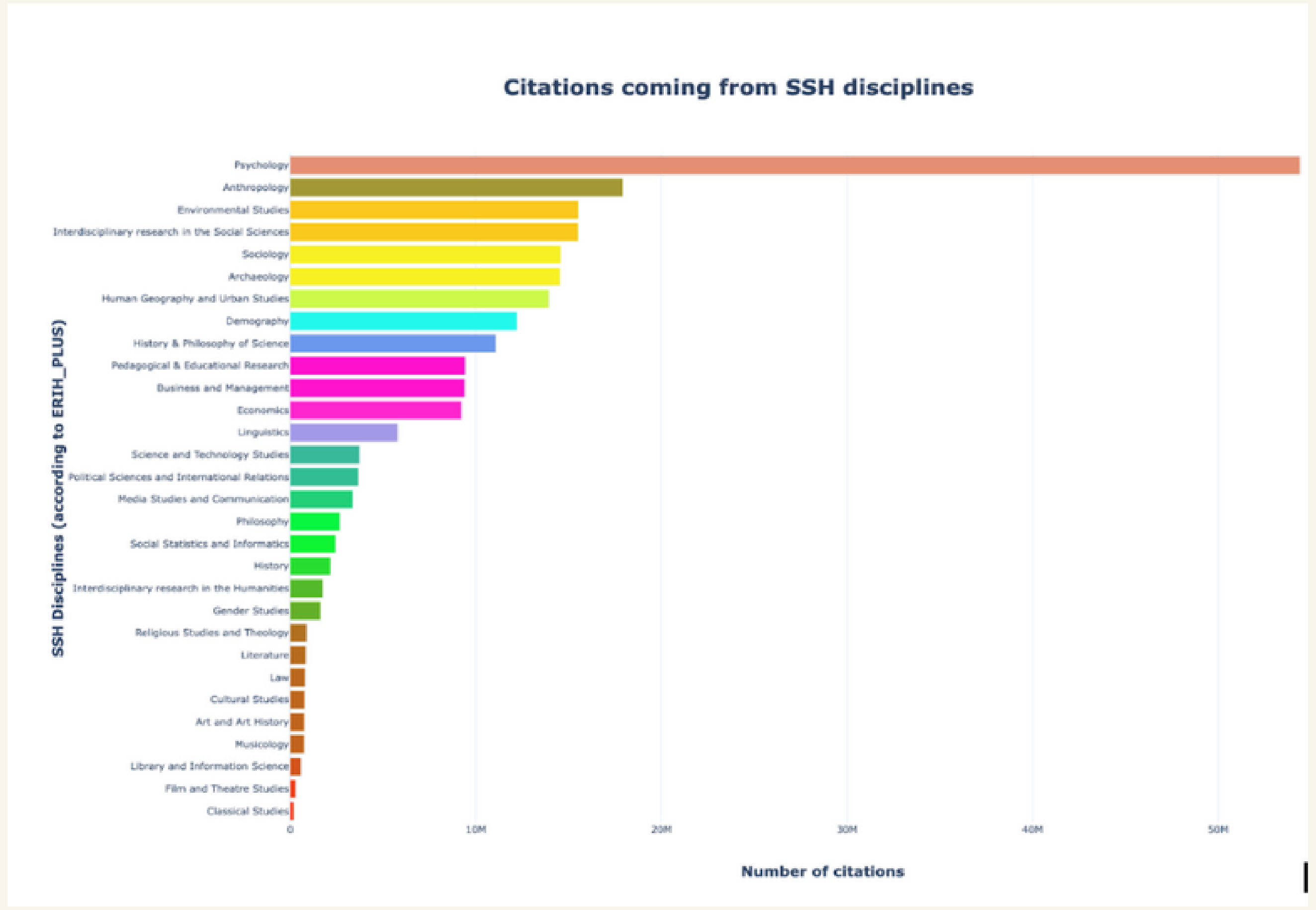


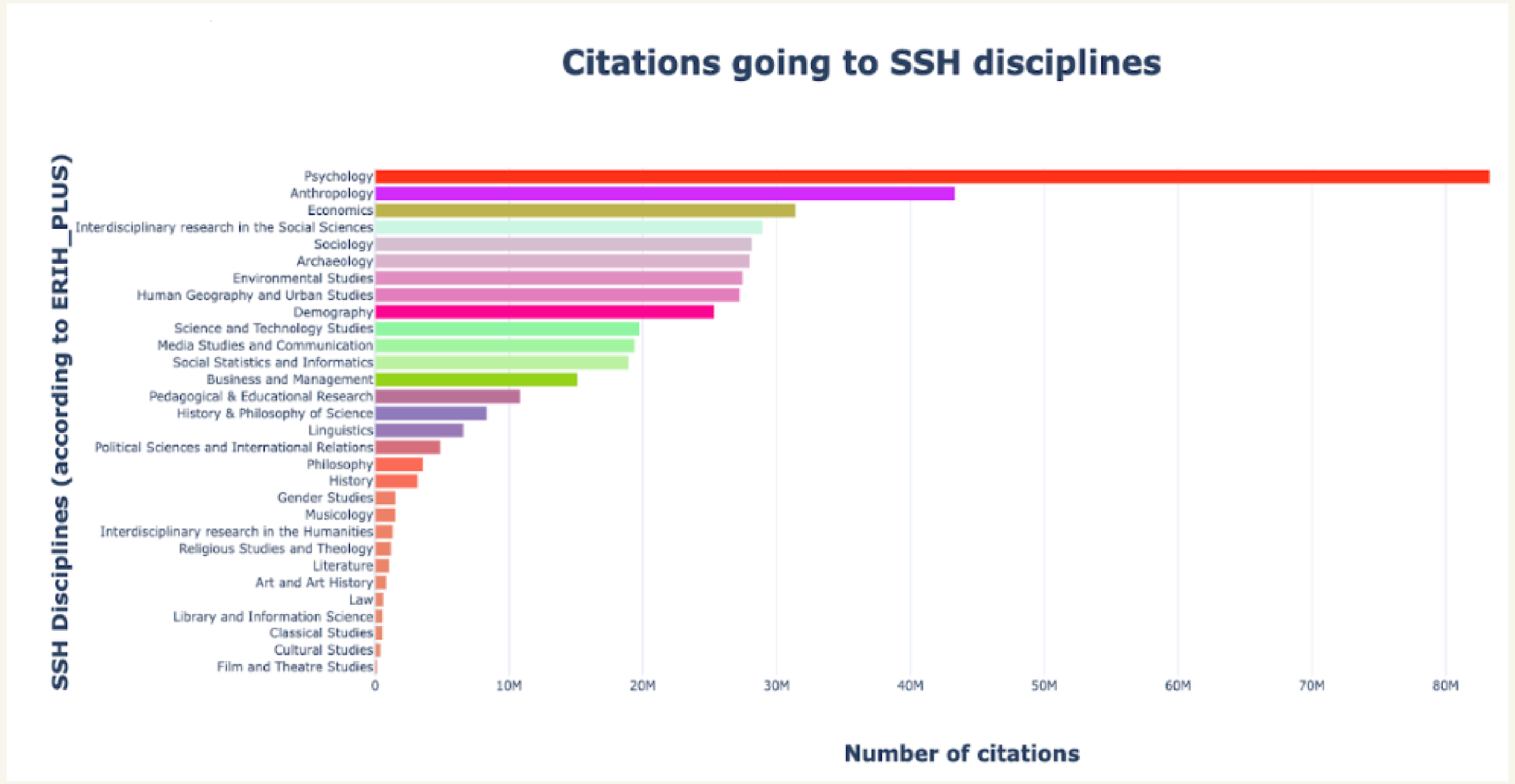
THE RESULTS

See our results here: https://opensciencepika.github.io/results/

ANSWER 2

The most cited and the most citing discipline in the field of Social Science and Humanities is **psychology**, having 54.512.160 citing DOIs and 83.291.583 cited DOIs.





THE RESULTS

See our results here: https://opensciencepika.github.io/results/

OUR FINDINGS

- Not all the DOIs included in COCI are also included in META - "*partial citations*"
- A citation may be totally or just partially included (or excluded) in the SSH field
- Publication years in which the greatest number of citations is recorded
- Why is Psychology the most citing and cited discipline?

THANK YOU FOR YOUR ATTENTION!