OAmonitor R package

Kristina Hettne[*] Barbara Vreede[†] Bianca Kramer[‡]

Last updated on 01 September, 2023

Introduction

This vignette explains the use of the OAmonitor package. We first go through the functionalities of the package and then provide an example for use.

Package overview

The OAmonitor takes publication data from a single year and determines per article its open access status, using the Directory of Open Access Journals, (DOAJ), Unpaywall and the association of universities in the Netherlands (VSNU) Open Access publishing deals (an excel sheet available through the VSNU). It uses peer reviewed journal articles registered in CRIS systems as input. It was created in a collaboration between Utrecht University Library and Leiden University Library.

OAmonitor basics

The OAmonitor has three basic parts: data gathering and cleaning, data classification and results reporting. Before user data can be loaded it needs to be in a specific format that is described in the "User data preparation" section.

User data preparation

User data is likely the export from a CRIS, or download from an online database (e.g. Scopus).

- It may consist of multiple independent source files.
- It must be tabular, and can be either csv (comma or semicolon-separated), tsv (tab-separated), xlsx or xls.

The workflow requires the following columns, which may or may not contain empty cells:

Column	Explanation	Empty cells?
Unique ID	source-specific ID	No
Organization unit	e.g. departments, faculties (but can be anything)	No
ISSN	max 2 columns (e.g. ISSN + eISSN)	Yes
DOI	digital object identifier (number or with http address before)	Yes

[*]Leiden University Libraries, https://orcid.org/0000-0002-4182-7560
[†]Utrecht University, https://orcid.org/0000-0002-5023-4601
[‡]Utrecht University, https://orcid.org/0000-0002-5965-6560

Column	Explanation	Empty cells?

Columns do not need to be renamed prior to use (but relevant columns do require a unique name).

User data configuration and loading

Download the excel template to configure where your input data files are located, what they are called and which columns that correspond to the ones that will be used by the script.

- Fill out the template
  - Put each individual data file in a new column
  - Fill out the name of each relevant column per data file
- Save the excel template directly in the project folder
- Place all input data files in a subfolder of the project folder (e.g. `data/`)

The data can be loaded into a combined data frame using the `open_everything()` function, where `file` is the name of the template file and `dir` is the folder where your input data files are stored.

```r
df <- open_everything(file = "FILENAME",
                      dir = "data")
```

External data gathering and cleaning

The following external data sources are used:

- VSNU (spreadsheet, available upon request)
- DOAJ (via API)
- Unpaywall (via API)

We recommend placing the VSNU excel file in the `data/` folder of your project. The DOAJ and UPW downloads will be placed in `data/clean/` automatically when generated so it is useful if you already created the folder `data/`; if that does not exist, the program will create it for you.

The VSNU data can be loaded by using the `get_vsnu()` function with the path to the VSNU file as argument.

```r
vsnudf <- get_vsnu("FILEPATH")
```

Note that running the DOAJ and Unpaywall APIs takes a while (i.e. several hours), so the function contains the option to save, and then reload the data at a later stage if needed.

Make sure that `save_results = T`, and fill out your own email address. We advice to select both lines and run them simultaneously, since they will take a while. Keep your computer open while the API code runs.

```r
get_doaj(df, source = "api", save_results = T)
get_upw(df, source = "api", email = "YOUREMAIL@UNI.NL", save_results = T)
```

When the API functions are done, the DOAJ and Unpaywall files can be loaded with the `get_doaj()` and `get_upw()` functions by providing the paths to the files as argument instead of `api`. These files are by default stored in `/data/clean`.

```r
doajdf <- get_doaj(df, source="FILEPATH")
upwdf <- get_upw(df, source="FILEPATH")
```

Data classification

If the argument `save_results` is set to TRUE, an output file containing the classification labels per article will be generated.

```r
df_class <- classify_oa(df,
                        doajdf=doajdf,
                        vsnudf=vsnudf,
                        upwdf=upwdf,
                        save_results = TRUE)
```

Results reporting

Results of the classification can be generated by either running the three functions `report_to_dataframe()`, `report_to_image()` and `report_to_alluvial()` separately and save the results or by running the `full_report()` function that runs the three functions and saves the results automatically. The `report_to_image()` and `report_to_alluvial()` can produce either proportional (argument `prop`) or absolute numbers (agrument `total`).

Note: Before running the `report_to_alluvial()` function, `StatStratum` must be set.

A full report includes:

- a summarizing table, reporting on percentages per organization unit
- a bar chart with total numbers
- a bar chart with proportional numbers
- an alluvial diagram for all papers (not deduplicated!)

All these are saved objects. You can find them in `output/` and `figures/`.

```r
report_to_dataframe(df_class)
report_to_image(df_class, type="total")
StatStratum <- ggalluvial::StatStratum
report_to_alluvial(df_class, type="prop")
full_report()
```

Quality check

For some faculties or departments, getting information based on ISSNs and DOIs will not be sufficient. To screen the data for information density, the function `check_info()` can be used.

```r
check_info(df_class, cutoff = 0.05, save=T)
```

`cutoff` is the maximum level of "NONE" allowed. If `save = TRUE`, the resulting data will be saved to `data/clean/`.

Custom reports

There are also a few ways to customize the reports, which we will elaborate on below.

Combinations of groups

To report on customized combinations of groups within your organization (e.g. all departments in a faculty, or all faculties) you can used the function `individual_reports()`. The output will be automatically saved in the folders `output/` and `figures/`.

- Download the excel template
- Fill out the template with sub-groups of organization units (spelled exactly like in org_unit!) that you want to include in a single report
- Give each list of sub-groups an appropriate title
- Save the filled out template in the project folder

```
individual_reports(df_class, path_report = "FILENAME")
```

Higher Education and Research Plan (HOOP) areas

A different kind of sub-report can be done by aggregating organization units and reporting on the headers. Here deduplication is done per aggregate. This can be used e.g. for HOOP-areas. The output will be automatically saved in the folders `output/` and `figures/`.

- Download the excel template
- Organize your organization units to match HOOP areas
- Save the filled out template in the project folder

```
hoop_report(df_class, path_hoop = "FILENAME")
```

Custom labels

After a manual screening, papers can be found to be accessible after all. Or, papers that are known to be in the repository, are not picked up by Unpaywall. While too much customization will affect how comparable results will be, it can be necessary to include these for internal reports. The classify_oa() function can be used to include custom labels for papers. Custom labels are only applied after all Gold and Hybrid classifications are given. Custom labels will only classify as Green.

- Download the excel template
- Edit the file by adding custom labels and listing the unique internal identifiers of papers

```
df_class_custom <- classify_oa(df, doajdf=doajdf, vsnudf=vsnudf, upwdf=upwdf,
                               custom=T, custom_path="FILEPATH")
```