Gathering Bibliometric Information using Scopus

by John Muschelli

Abstract An abstract of less than 150 words.

Introduction

Scopus a repository of information about scientific articles and books, which includes information about authors, citations, and abstracts of these pieces of literature. Scopus claims to have the largest database of this information.

One common task for researchers is to keep the curriculum vitae (CV) up to date. That requires having up to date information on the papers published and under submission. Keeping track of these papers can be tedious and automated solutions could exist, but are not widely used (THINK OF SERVICES THAT TRY TO do this). One conern is always to be missing certain crucial papers in your CV. Although **rscopus** does not provide these tools specifically, it can be used to consistently cross-reference information about publications with a CV.

Additionally on CVs, it is useful to present information of the impact of a paper on the CV. This can be done by highlighting certain pieces of information, such is done in NIH biosketches, or ranking them based on some metric. The metric commonly used is the number of citations. Also, information about the journal impact factor may be taken into account. We do not note that these are particularly good metrics or metrics that reflect true impact, but are simply those that we have seen used in practice.

rscopus allows you to interface with Scopus APIs and gather information about authors, affiliations, articles, and abstracts. The **bibliometrix** package provides a level of integration that is useful for using multiple packages that deal with bibliometric data, incorporating functionality from **rscopus**. The **bibliometrix** package also enables users to analyze data from ISI Web of Knowledge (WoK) and PubMed. Web of Knowledge is one competitor to Scopus, but **bibliometrix** does not have an interface to the WoK API; therefore data must be manually exported from the site into `R`. Additional access to the web of Science API would be useful and has been implemented in a GitHub package **rwos** (`https://github.com/juba/rwos`), but is not on CRAN.

Moreover, other packages such as **scholar** and **gcite** can provide interfaces to the Google Scholar citation information. Using these in combination with **rscopus** can more likely guarantee complete information.

As compared to Google Scholar, WoK and Scopus information is **more curated** (is that true?).

As many pieces of academic promotion are related citation metrics or impact, these metrics can have real-life implications.

Here we will present a few examples of how to use the **rscopus** package, and present a simple descriptive analysis of citations and citation indices from people within the same field.

API Key

Before using the package, one must obtain an access key to the API from `https://dev.elsevier.com/sc_apis.html` with the following steps:

1. Go to `https://dev.elsevier.com/user/login`. Login or create a free account.
2. Click "Create API Key". Put in a label, such as `rscopus key`. Add a website. `http://example.com` is fine if you do not have a site.
3. **Read** and agree to the terms of service if you do indeed agree.
4. Add `Elsevier_API = "API KEY GOES HERE"` to `~/.Renviron` file, or add `export Elsevier_API=API KEY GOES HERE` to your `~/.bash_profile`.

Alternatively, you you can either set the API key using `rscopus::set_api_key` or by `options("elsevier_api_key" = api_key)`. You can access the API key using `rscopus::get_api_key`.

You should be able to test out the API key using the interactive Scopus APIs.

A note about API keys and IP addresses

The API Key is bound to a set of IP addresses, usually bound to your institution or organization (see https://dev.elsevier.com/tecdoc_api_authentication.html). Therefore, if you are using **rscopus** for a Shiny application, you must host the Shiny application from the institution/organization servers in some way. Also, you cannot access the Scopus API with this key if you are offsite and must VPN into the server or use a computing cluster with an institution IP.

Use cases

```
library(rscopus)
author_info = author_retrieval(last_name = "Muschelli", first_name = "J")

#> HTTP specified is (without API key): https://api.elsevier.com/content/search/author?query=AUTHFIRST%28J%29

#> Authors found:

#>        auth_name       au_id affil_id
#> 1 John Muschelli 40462056100 60006183
#>                                       affil_name
#> 1 Johns Hopkins Bloomberg School of Public Health

#> HTTP specified is:https://api.elsevier.com/content/author/author_id/40462056100

names(author_info$content)

#> [1] "author-retrieval-response"
```

In many cases with the API, you will specify the author identifier (`au_id`) instead of a first and last name, as there may be many authors with the same name. This identifier is unique to this author, though curation errors do happen and someone may have 2 unique identifiers. These identifiers can be merged by request on the Scopus website. In order to get the identifier from Scopus, you can search using a first and last name using the `process_author_name` command.

```
auth_name = process_author_name(last_name = "Muschelli", first_name = "John")

#> HTTP specified is (without API key): https://api.elsevier.com/content/search/author?query=AUTHFIRST%28John

#> Authors found:

#>        auth_name       au_id affil_id
#> 1 John Muschelli 40462056100 60006183
#>                                       affil_name
#> 1 Johns Hopkins Bloomberg School of Public Health

auth_name

#> $first_name
#> [1] "John"
#>
#> $last_name
#> [1] "Muschelli"
#>
#> $au_id
#> [1] "40462056100"
```

Retrieving information about an author

Most times, yo udo not have

```
auth_info = get_author_info(last_name = "Smith", first_name = "S")

#> HTTP specified is (without API key): https://api.elsevier.com/content/search/author?query=AUTHFIRST%28S%29

head(auth_info, 5)
```

```
#>                    auth_name       au_id affil_id
#> 1          Alfred J.S. Smith  7406754790 60003269
#> 2             Peter J. Smith  7406996870 60002316
#> 3           Peter J.S. Smith 55723644900 60015875
#> 4 Priscilla S. Kincaid-Smith 35805496900 60026553
#> 5          Sidney C.C. Smith  7406657586 60025111
#>                                      affil_name
#> 1                            Princeton University
#> 2               Victoria University of Wellington
#> 3                          University of Aberdeen
#> 4                         University of Melbourne
#> 5 The University of North Carolina at Chapel Hill

ssmith = process_author_name(last_name = "Smith", first_name = "S")

#> HTTP specified is (without API key): https://api.elsevier.com/content/search/author?query=AUTHFIRST%28S%29

#> Authors found:

#>           auth_name      au_id affil_id           affil_name
#> 1 Alfred J.S. Smith 7406754790 60003269 Princeton University

ssmith

#> $first_name
#> [1] "S"
#>
#> $last_name
#> [1] "Smith"
#>
#> $au_id
#> [1] "7406754790"
```

This section may contain a figure such as Figure 1.

Figure 1: The logo of R.

Another section

There will likely be several sections, perhaps including code snippets, such as:

```
x <- 1:10
x

#>  [1]  1  2  3  4  5  6  7  8  9 10
```

Summary

This file is only a basic article template. For full details of *The R Journal* style and information on how to prepare your article for submission, see the Instructions for Authors.

John Muschelli
Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health
615 N Wolfe St Baltimore, MD 21205
jmuschel@jhsph.edu