Gathering Bibliometric Information from the Scopus API using rscopus

by John Muschelli

Abstract We demonstrate how to download author and affiliation using the `rscopus` package, interacting with the Elsevier Scopus API. We demonstrate how to manipulate the output from the API into organized data that can be analyzed. We present options on how to obtain the number of citations from an author and use that to calculate citation metrics with the output. We also show how to download additional information from an article such as the figures and supplemental material.

Introduction

Elsevier provides a number of APIs (application program interfaces) to extract information about scholarly publications (`https://dev.elsevier.com/`). This information is can provide insights into how research is done and provide the data to perform experiments and analyses of metascience. Elsevier's Scopus a repository of information about articles, authors, and institutions. Scopus claims to have the largest database of this information (`https://www.elsevier.com/solutions/scopus`). Therefore, providing users an interface to this repository should be worthwhile.

One common task for researchers is to keep his or her curriculum vitae (CV) up to date. That requires having accurate information on the papers published and under submission. Keeping track of these papers can be tedious and solutions could exist if one could retrieve papers automatically. One concern is missing certain crucial papers in your CV. Although **rscopus** does not provide these tools specifically, it provides tools to download publications about an author, which can be be used to consistently cross-reference information about publications with a CV.

Additionally on CVs, one may present information of the impact of a paper. This can be done by highlighting certain pieces of information, such is done in NIH biosketches, or ranking papers based on some metric. As promotions and grant review may be affected by highlighting impact, these metrics can have real-life implications. One metric commonly used is the number of citations. Additionally, information about the journal impact factor may be shown. We do not imply that these are particularly good metrics or metrics that reflect true impact, but are simply those that we have seen used in practice.

The **rscopus** package allows users to interface with Scopus APIs and gather information about authors, affiliations, articles, and abstracts. Currently in R, packages exist for bibliometric analysis, but commonly require the data to be downloaded from a website or online interface. For example, the **bibliometrix** (Aria and Cuccurullo, 2017) 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 and has not been updated recently (over 1 year). We have also created an interface with the Web of Science APIs (`https://github.com/muschellij2/webofscience`), but have not been given access to the APIs to test them as it is still in beta.

As compared to Google Scholar, Scopus and WoK claim the information from these sources is more curated. Other packages such as **scholar** (Keirstead, 2016) and **gcite** (Muschelli, 2018) can provide interfaces to the Google Scholar citation information. Using these in combination with **rscopus** can more likely guarantee complete information. Also, comparing these different repositories can demonstrate biases or differences in the number of citations across the different platforms. Also, Scopus has a true API compared to the packages above that extract information from the Google Scholar webpage, which will likely be more stable, more robust to API changes, and be within the terms of service for the user.

Scopus has a number of APIs available (`https://dev.elsevier.com/sc_apis.html`). Here we will present examples of how to use the **rscopus** package, including: searching for authors or affiliations, calculating citation indices for an author, retrieving abstract information on articles, and downloading artifacts from an article.

API Key

Before using the package, one must obtain an access key to the API from Elsevier 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. You should be able to test out the API key using the interactive Scopus APIs.
5. 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, which will implicitly set options("elsevier_api_key" = api_key). You can access the API key using rscopus::get_api_key.

Once you have an API key set up, you can access the key using get_api_key, which will check multiple places for the presence of the key:

```
library(rscopus)
key = get_api_key()
```

As you may want to hide this key from documents, the default print options do not display it:

```
key
```

```
<hidden api key, use print(, reveal = TRUE) to see it>
```

There are other helpful functions such as have_api_key, which tests if an API key has been specified:

```
have_api_key()
```

```
[1] TRUE
```

In each function to an API endpoint, you may pass in the API key as the argument api_key as well, rather than specifically look in the ~/.Renviron or options().

A note about API keys and IP addresses

The API Key is bound to a set of IP addresses, usually from 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.

Methods: Use cases

In this section we will present some use cases for the API, focusing on retrieving information about authors and affiliations.

Processing author names to identifiers

Researchers commonly would like to gather information about a set of authors. Most times the authors are the given by first and last names or initials; additional information such as affiliation may be available. Scopus provides unique identifier for authors (au_id) or affiliations (affil_id), amongst others. 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. In order to get the author identifier from Scopus, you can search using a first and last name using the process_author_name command. For example, let us try to identify the author ID for John Muschelli:

```
auth_info = process_author_name(last_name = "Muschelli", first_name = "John",
                                verbose = FALSE)
auth_info
```

```
$au_id
[1] "40462056100"

$first_name
[1] "John"

$last_name
[1] "Muschelli"
```

The output is a simple list of first and last name with an author ID. The function chooses the first author found, which may be useful if the author name is somewhat unique. We will show below how to search when the name is not as unique. This ID 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.

Retrieving author citation data

In order to get data about papers and citations from an author, the author_data function will retrieve this information:

```
jm = author_data(last_name = "Muschelli", first_name = "John", verbose = FALSE)
```

We see the output is a list of the converted entries from the JSON output Scopus API, a data.frame of the results for citations, and a list named full_data:

```
class(jm)
```

```
[1] "list"
```

```
names(jm)
```

```
[1] "entries"    "df"         "first_name" "last_name"  "au_id"
[6] "full_data"
```

The data.frame df has the information many users wish to retrieve, which is information about the author documents such as the Scopus ID for that paper (dc:identifier), the title of the paper (dc:title), and the number of citations (citedby-count). Here we present short_title, first 3 "relevant" words of the title (see unique_title in supplemental material), instead of the full document title from dc:title for viewing purposes as titles can be quite long.

```
jm$df$short_title = unique_title(jm$df$`dc:title`)
head(jm$df[, c("dc:identifier", "short_title", "citedby-count")])
```

```
          dc:identifier                          short_title
1 SCOPUS_ID:85053246791         Objective Evaluation Multiple
2 SCOPUS_ID:85043338865              MIMoSA: Automated Method
3 SCOPUS_ID:85047750078           Radiomic subtyping improves
4 SCOPUS_ID:85028874240       Feasibility Coping Effectiveness
5 SCOPUS_ID:85050271095      Freesurfer: Connecting Freesurfer
6 SCOPUS_ID:85009266881 Thrombolytic removal intraventricular
  citedby-count
1             0
2             1
3             3
4             2
5             0
6            49
```

We see that the full_data has the object df duplicated in it, as well as other data.frames:

```
names(jm$full_data)
```

```
[1] "df"          "affiliation" "author"
```

These data.frames can have additional information about co-author affiliations or co-author information. This information may be useful for creating network graphs. For example, to get all authors from all the papers, you can use the author element from full_data:

```
head(jm$full_data$author[, c("authid", "authname", "surname", "afid.$", "entry_number")])
```

```
       authid     authname   surname   afid.$ entry_number
1  8431704700 Commowick O. Commowick 60030553            1
2 57203861434    Istace A.    Istace 60001780            1
3 57199507814      Kain M.      Kain 60030553            1
4 57197801981   Laurent B.   Laurent 60105610            1
5 57203867656     Leray F.     Leray 60030553            1
6 57203864793     Simon M.     Simon 60030553            1
```

The column entry_number indicates which element this information came in the entry list (retrieved from content from **httr** (Wickham, 2017), which calls fromJSON from **jsonlite** (Ooms, 2014)). This column should merge with the data.frame of citations, as well as the information about author affiliations, which is located in the affiliation data.frame from full_data:

```
head(jm$full_data$affiliation[, c("afid", "affiliation-country", "entry_number",  "affilname")])
```

```
      afid affiliation-country entry_number
1 60030553              France            1
2 60001780              France            1
3 60105610              France            1
4 60062760              France            1
5 60028893              France            1
6 60028893              France            1
                                                              affilname
1                                                   Universite de Rennes 1
2                                                 Centre Hospitalier Lyon-Sud
3                      Laboratoire de Traitement de l'Information Medicale
4 Centre de Recherche en Acquisition et Traitement d'Images pour la Sante
5                               Centre Hospitalier Universitaire de Rennes
6                               Centre Hospitalier Universitaire de Rennes
```

This information is rich for understanding information about an author's publication record, how many citations are recorded for a specific article, which journals have been published in, and who has co-authored publications with an author. Here we rank the authors based on number of co-authored publications. We will use **dplyr** (Wickham et al., 2018) for data manipulation:

```
library(dplyr)
au_id = unique(jm$df$au_id)
co_authors = jm$full_data$author %>%
  filter(!authid %in% au_id)
co_authors = co_authors %>%
  add_count(authid) %>%
  select(n, authid, authname, surname) %>%
  distinct() %>% arrange(-n)
head(co_authors)
```

```
# A tibble: 6 x 4
      n authid      authname       surname
  <int> <chr>       <chr>          <chr>
1    18 7003584880  Mostofsky S.   Mostofsky
2    16 7003771238  Pekar J.       Pekar
3    12 7103268788  Hanley D.      Hanley
4    11 55587557800 Sweeney E.     Sweeney
5    10 8582425200  Barber A.      Barber
6     9 6603085071  Crainiceanu C. Crainiceanu
```

More analyses of co-authorship can be done, such as determining how many international collaborators an author had on papers based on affiliation.

Retrieving summary information about an author

The author_retrieval function can gather summary information about an author using the author identifier or name.

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

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

```
names(author_info$content)
```

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

```
class(author_info$content$`author-retrieval-response`)
```

```
[1] "list"
```

In the standard output from the Scopus API after conversion in **httr**, there are elements of the list named entries or entry. The low-level function gen_entries_to_df attempts to coerce this list into a standard data.frame for more usability, but may not perform perfectly as lists from JSON cannot always be directly coerced into a rectangular format. For example, here we will convert that output into a data.frame:

```
gen_entries_to_df(author_info$content$`author-retrieval-response`)$df
```

```
  @status @_fa
1   found true
                                            coredata.prism:url
1 http://api.elsevier.com/content/author/author_id/40462056100
  coredata.dc:identifier       coredata.eid      coredata.orcid
1  AUTHOR_ID:40462056100 9-s2.0-40462056100 0000-0001-6469-1750
  coredata.document-count coredata.cited-by-count coredata.citation-count
1                      36                     681                     804
                                                                             coredata.link.@href
1 https://www.scopus.com/authid/detail.uri?partnerID=HzOxMe3b&authorId=40462056100&origin=inward
  coredata.link.@rel coredata.link.@_fa preferred-name.surname
1      scopus-author               true              Muschelli
  preferred-name.given-name preferred-name.initials
1                      John                      J.
             affiliation-current.affiliation-name
1 Johns Hopkins Bloomberg School of Public Health
  affiliation-current.affiliation-city
1                            Baltimore
  affiliation-current.affiliation-country publication-range.start
1                           United States                    2011
  publication-range.end entry_number
1                  2018            1
```

but this list from author_retrieval typically only has one element, and may be easily referenced using $ as a list. Most of the functions implicitly call gen_entries_to_df to convert the data when available, and additional organizing functions exist such as process_complete_multi_author_info discussed below. Overall, author_retrieval gives general info about an author, but this information and more can likely be extracted using the author_data function above.

Calculating author indices

With the data from the author_data output (jm$df), we can calculate citation indices. Here we calculate the overall *h*-index (Hirsch, 2005):

```
h_data = jm$df %>%
  mutate(citations = as.numeric(`citedby-count`)) %>%
  arrange(-citations) %>%
  mutate(n_papers = 1:n())
head(h_data[, c("short_title", "citations", "n_papers")])
```

```
                             short_title citations n_papers
1             Minimally invasive surgery       141        1
2     Disruption functional organization        68        2
3     Reduction motion-related artifacts        63        3
4          Minimally invasive evacuation        54        4
5 Resolution intraventricular hemorrhage        50        5
6  Thrombolytic removal intraventricular        49        6
```

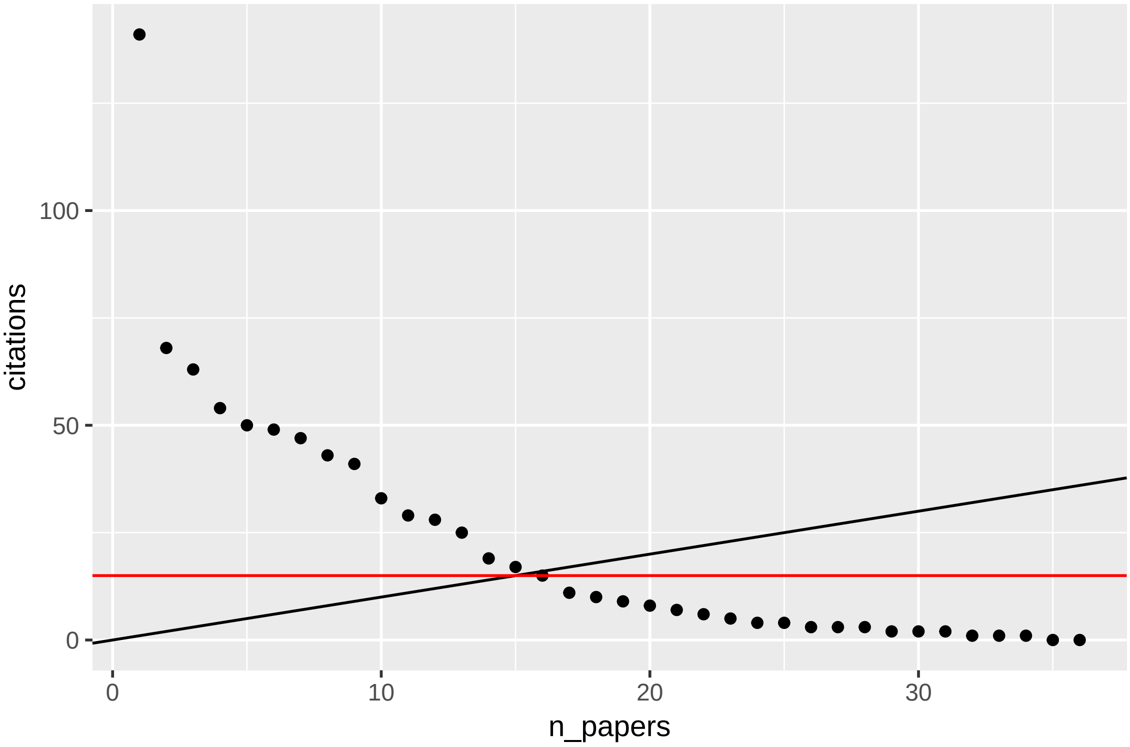


Figure 1: Calculating an h-index. Here we plot the number of papers versus the number of citations for that paper. This plot is the basis for the h-index. The X=Y line is displayed in black and the red line is where the curve passes the X=Y line, which is the h-index, a value of 15.

```
h_index = max(which(h_data$citations >= h_data$n_papers))
cat(paste0("Calculated h-index is ", h_index))

Calculated h-index is 15
```

Using **ggplot2** (Wickham, 2016), we can also visually show the *h*-index computation, where we plot the number of citations versus the number of papers (cumulatively) along with the X-Y line:

```
library(ggplot2)
h_data %>%
  ggplot(aes(x = n_papers, y = citations)) +
  geom_point() + geom_abline(slope = 1, intercept = 0) +
  geom_hline(yintercept = h_index, color = "red")
```

Additional indices can be created from the data, such as the *g*-index (Egghe, 2006):

```
h_data = h_data %>% mutate(sum_citations = cumsum(citations))
g_index = max(which(h_data$sum_citations >= h_data$n_papers^2))
cat(paste0("Calculated g-index is ", g_index))

Calculated g-index is 28
```

We could create a series of metrics based on this data and compare it to or supplement other metrics of performance. Note, however, longitudinal information is lacking on the number of citations per year. We will discuss citation information over time in Section 2.2.7.

Retrieving information about an author

In `process_author_name`, we demonstrated how to get information form an author with a relatively unique name. If this is not the case, the `get_complete_author_info`, which is the backend function for `process_author_name`, can present more results. In order to retrieve author IDs from first and last names, the `get_complete_author_info` can be used. Here we search for authors with the last name West and first initial M:

```
last_name = "West"
first_name = "M"
auth_info_list = get_complete_author_info(last_name = last_name, first_name = first_name)
class(auth_info_list)
```

```
[1] "list"
```

```
names(auth_info_list)
```

```
[1] "get_statement" "content"
```

We see here, which is common in in some low-level functions returned from the rscopus API, the output is a list with elements get_statement, which returns an object of class response (from the **httr** package), and content, which is the content from the response. Most times, the content is of interest, but failed requests may be explored with the get_statement output for debugging.

Again, we can use gen_entries_to_df convert that output into a data.frame, which is more manageable. Here show see the author information retrieved from the call:

```
coerced = gen_entries_to_df(auth_info_list$content$`search-results`$entry)
names(coerced)
```

```
[1] "df"           "name-variant" "subject-area"
```

```
head(coerced$df[, c("dc:identifier",  "preferred-name.surname",
                    "preferred-name.given-name", "affiliation-current.affiliation-name" )])
```

```
          dc:identifier preferred-name.surname preferred-name.given-name
1 AUTHOR_ID:35480328200                   West                Catharine
2 AUTHOR_ID:35419377800                   West               Malcolm J.
3  AUTHOR_ID:7003392768            Diener-West                    Marie
4  AUTHOR_ID:7402395730                   West                Robert M.
5  AUTHOR_ID:7402068812                   West          Michael Abigail
6  AUTHOR_ID:7401998578                   West                 David M.
             affiliation-current.affiliation-name
1                        University of Manchester
2                James Cook University, Australia
3 Johns Hopkins Bloomberg School of Public Health
4                             University of Leeds
5         University of Pittsburgh Medical Center
6                               Massey University
```

We see this has information about the multiple authors returned, along with names, variations on those names, number of documents, and affiliations. We can then extract the author ID we want from this data.frame. This process is wrapped in the get_author_info:

```
auth_info_df = get_author_info(last_name = last_name,
                              first_name = first_name)
head(auth_info_df)
```

```
             auth_name       au_id          affil_id
1       Catharine West 35480328200          60003771
2      Malcolm J. West 35419377800          60019870
3    Marie Diener-West  7003392768          60006183
4       Robert M. West  7402395730          60012070
5 Michael Abigail West  7402068812 60012018 60023691
6        David M. West  7401998578          60008221
                                       affil_name
1                        University of Manchester
2                James Cook University, Australia
3 Johns Hopkins Bloomberg School of Public Health
4                             University of Leeds
5         University of Pittsburgh Medical Center
6                               Massey University
```

We should note this information is condensed and a subset that is available from get_complete_author_info, also with more standardized column names. Now, one could use the au_id retrieved from this output for searching and retrieval.

If you had an affiliation ID, such as 60006183 for the Johns Hopkins Bloomberg School of Public Health, we can pass this to get_author_info or process_author_name:

```
spec_affil = get_author_info(
  last_name = last_name,
  first_name = first_name,
  affil_id = 60006183)
spec_affil
```

```
          auth_name      au_id affil_id
1 Marie Diener-West 7003392768 60006183
                                        affil_name
1 Johns Hopkins Bloomberg School of Public Health
```

Thus, we can use the combination of the names of the author and affiliation information. This may be useful when searching multiple authors at the same institution. We could also pass in affil_name instead of affil_id to get_author_info, but the name must be extremely specific for institution as the function will call get_affiliation_info, which is discussed below.

Retrieving information about multiple authors

In order to get information from multiple authors, one could loop over author information, but this is inefficient for code and API calls. The complete_multi_author_info function can perform this operation. One caveat is that it requires author identifiers and not names. We can take the author IDs from auth_info_df to retrieve information for all these authors:

```
all_author_info = complete_multi_author_info(au_id = auth_info_df$au_id)
names(all_author_info)
```

```
[1] "get_statement" "content"       "au_id"
```

This result is again a low-level output from the API. We can use the process_complete_multi_author_info function to process this into a more amenable solution:

```
processed = process_complete_multi_author_info(all_author_info)
head(names(processed))
```

```
[1] "35480328200" "35419377800" "7003392768"  "7402395730"  "7402068812"
[6] "7401998578"
```

Now, each element is the author ID, which contains a list of data.frames.

For example, for the ID 35480328200, we can see the elements for that specific author.

```
names(processed$`35480328200`)
```

```
[1] "info"                "links"               "other_names"
[4] "subject_areas"       "affiliation_history" "journals"
```

We can then access the affiliation history, which includes information such as department and university:

```
head(processed$`35480328200`$affiliation_history[, c("ip-doc.afdispname", "ip-doc.id", "ip-doc.type")], 3)
```

```
                                        ip-doc.afdispname ip-doc.id
1 University of Manchester, Radiotherapy Related Research 110273198
2   University of Manchester, Division of Cancer Sciences 119995430
3   University of Manchester, Centre for Imaging Sciences 113240010
  ip-doc.type
1        dept
2        dept
3        dept
```

The multi_author_info will perform both of these operations together. This result is still not "tidy" (Wickham et al., 2014) in many respects, but parts can be combined using **purrr** (Henry and Wickham, 2018):

```
journals = purrr::map_df(processed, `$`, "journals", .id = "au_id")
head(journals)
```

```
        au_id type                                        sourcetitle
1 35480328200    j                                     Cancer Letters
2 35480328200    j   Journal of Cancer Research and Clinical Oncology
3 35480328200    j                               Nature Reviews Cancer
4 35480328200    j Annals of the Royal College of Surgeons of England
5 35480328200    j                                     Cancer Letters
6 35480328200    j                         PLoS Computational Biology
           sourcetitle-abbrev     issn
1                Cancer Lett. 03043835
2 J. Cancer Res. Clin. Oncol. 01715216
3            Nat. Rev. Cancer 1474175X
4   Ann. R. Coll. Surg. Engl. 00358843
5                Cancer Lett. 18727980
6          PLoS Comput. Biol. 1553734X
```

Thus, in the above code we could compare the journals each author published in compared to the other authors in the list.

Citations over time

Some APIs from Elsevier are disabled by default (see https://dev.elsevier.com/api_key_settings.html). Notably, the Citations Overview API is disabled, which allows users to access information about citations over time for articles of authors. This citation information is particularly useful for creating bibliometric indices that rely on subsetting data based on minimum or maximum year or calculating time-dependent metrics. The rscopus package interfaces with these APIs, but the API must be enabled for that specific API key. Here, we show the output of the failed request as it may be useful for debugging:

```
sc_id = jm$df$`dc:identifier`[1]
cit_try = citation_retrieval(scopus_id = sc_id)
httr::status_code(cit_try$get_statement)
```

```
[1] 403
```

```
cit_try$content
```

```
$`service-error`
$`service-error`$status
$`service-error`$status$statusCode
[1] "AUTHENTICATION_ERROR"

$`service-error`$status$statusText
[1] "Requestor configuration settings insufficient for access to this resource."
```

On the Scopus website, however, one can searching for authors, select up to 15 authors, and then create a “Citation Overview”, which will give this citation information, and export it in a CSV format. The **rscopus** package provides a read_cto function to read in this data.

We also provide an example export from a single author in the package:

```
file = system.file("extdata", "CTOExport.csv", package = "rscopus")
citations_over_time = rscopus::read_cto(file)
names(citations_over_time)
```

```
[1] "data"               "year_columns"       "author_information"
```

The real information is in the data element of this list, which contains the yearly citation information (again titles have been shortened):

```
yr_cols = citations_over_time$year_columns
citations_over_time = citations_over_time$data
citations_over_time = citations_over_time %>%
  mutate(short_title = unique_title(`Document Title`))
head(citations_over_time[, c("short_title", yr_cols[1:5])])
```

```
                            short_title <2008 2008 2009 2010 2011
1        Objective Evaluation Multiple     0    0    0    0    0
2             MIMoSA: Automated Method     0    0    0    0    0
3          Radiomic subtyping improves     0    0    0    0    0
4      Feasibility Coping Effectiveness     0    0    0    0    0
5     Freesurfer: Connecting Freesurfer     0    0    0    0    0
6 Thrombolytic removal intraventricular     0    0    0    0    0
```

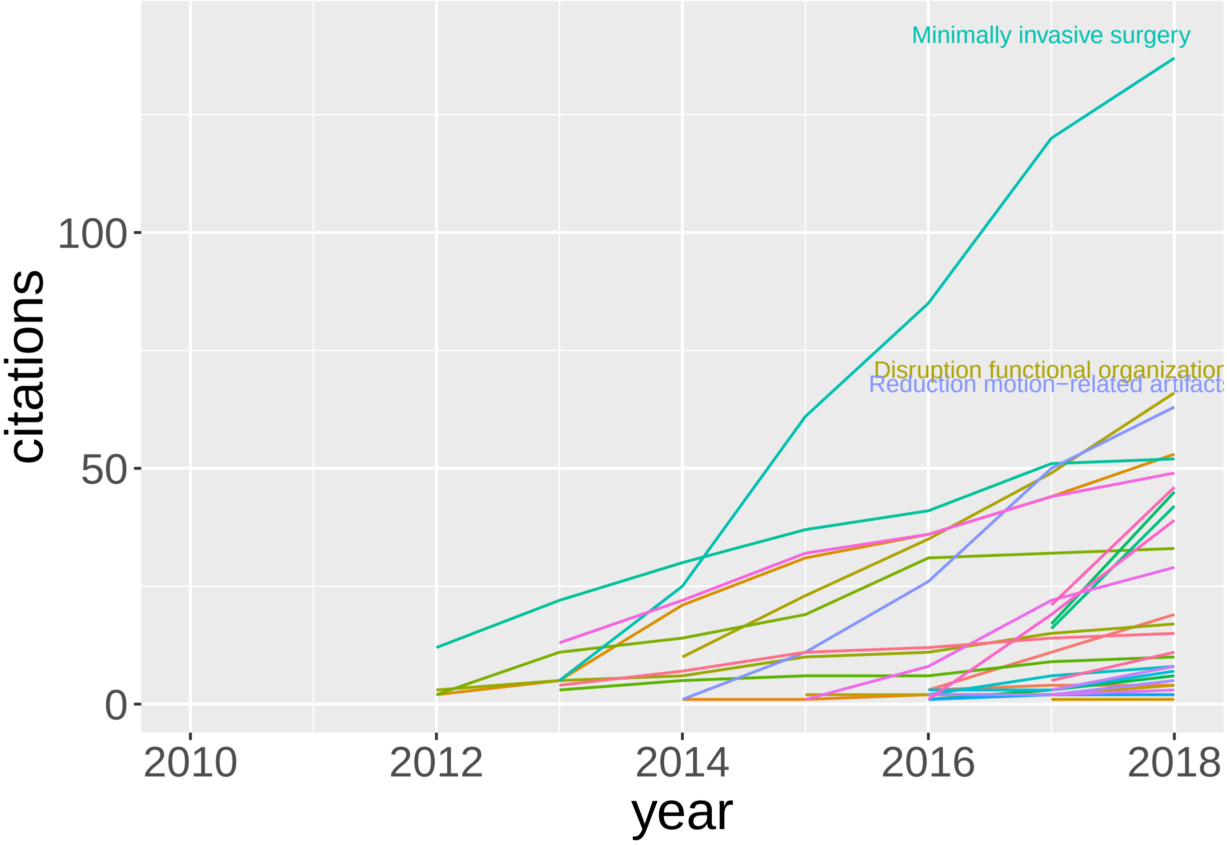
In the citation overview, you must specify a range of years on Scopus, with a maximum of 15 years. As many times this wide format is not what you want to plot, or in a "tidy" format, the **rscopus** helper function read_cto_long will read the data in long format, done by **tidyr** (Wickham and Henry, 2018). Here we use **dplyr** to arrange the data by maximum number of citations per year:

```
long_cite = rscopus::read_cto_long(file)
long_cite = long_cite$data %>%
  group_by(`Document Title`, year) %>% # get the citations per year
  summarize(citations = sum(citations), # aggregate - some duplicates merged
           `Publication Year` = unique(`Publication Year`)) %>% # keep the year in data
  mutate(short_title = unique_title(`Document Title`))
long_cite = long_cite %>% arrange(-citations, year, short_title)
head(long_cite[, c("short_title", "year", "citations")])
```

```
# A tibble: 6 x 3
  short_title                           year  citations
  <chr>                                 <fct>     <int>
1 Minimally invasive surgery            2015         36
2 Minimally invasive surgery            2017         35
3 ISLES 2015 public                     2018         28
4 Large-scale radiomic profiling        2018         26
5 Thrombolytic removal intraventricular 2018         25
6 Minimally invasive surgery            2016         24
```

Thus, we have one record per year and article. Here we will plot the cumulative citations per each paper over the years of publication and label the top 3 cited papers:

```
# get cumulative sum
csum = long_cite %>%
  # any missing data had no citations
  mutate(citations = ifelse(is.na(citations), 0, citations)) %>%
  arrange(`Document Title`, year) %>% # sort for cumsum
  group_by(`Document Title`) %>%
  mutate(citations = cumsum(citations))
# remove past and future with as.integer
csum = csum %>%
  mutate(year = as.integer(as.character(year))) %>%
  filter(!is.na(year)) %>% # remove < 2008 and > 2018 years
  filter(year >= `Publication Year` & citations > 0) # keep only relevant data for paper
# grab last citations and top 3 papers
last_year = csum %>%
  arrange(`Document Title`, year) %>% # sort for slice later
  group_by(`Document Title`) %>%
  slice(n()) %>% # keep last as max citations
  ungroup %>% arrange(-citations) %>%
  head(3)  # get top 3
g = ggplot(csum,
          aes(x = year, y = citations, color = short_title  )) +
  xlim(c(2010, 2018)) + geom_line() +
  # label the titles numbers for top 3
  geom_text(data = last_year, size = 3, aes(label = short_title),
           nudge_x = -1, nudge_y = 5)
# don't want label for document title - too many entries
g + guides(color = FALSE) + theme(text = element_text(size = 20))
```



Thus, we can present visually how the number of citations has changed and may be able to highlight which papers are gaining or waning in citations over time. This plot may not provide deep insights, but the same plot could be made by scientific sub-fields or for specific journals, which may indicate trends in published articles for example.

Retrieving affiliation information

In order to get information about an affiliation, the get_affiliation_info can be used. Here we will look for the pattern Johns Hopkins:

```
jhu_info = get_affiliation_info(affil_name = "Johns Hopkins")
head(jhu_info[, c("affil_id", "affil_name")])
```

```
  affil_id                                         affil_name
1 60005248                            Johns Hopkins University
2 60001117                The Johns Hopkins School of Medicine
3 60006183          Johns Hopkins Bloomberg School of Public Health
4 60001555                              Johns Hopkins Hospital
5 60003443                      Johns Hopkins Medical Institutions
6 60022054 The Johns Hopkins University Applied Physics Laboratory
```

This function implicitly calls affil_search, a lower-level function which searches the affiliation information from Scopus. Additional information can be extracted using affil_search, but this typically includes a large number of records as it searches all the documents. This affiliation ID again can be used to be more specific when searching authors or documents. Thus, we see that if we had only used "Johns Hopkins" as the query and not "Johns Hopkins Bloomberg School of Public Health", we would get back a different top candidate. This specificity is why we recommend using affiliation ID if using affiliation information in get_author_info.

Searching articles by abstract

In some cases, one may have an article in mind and would like information about the authors of that article In order to get the author IDs from the article identifier, one can use the abstract_retrieval function. There are multiple identifiers that can be used, such as PubMed ID, DOI, and we will use the Scopus ID as that is what is returned from the author_data output:

```
sc_id = jm$df$`dc:identifier`[1]
# retrieve abstract
abstract = abstract_retrieval(id = sc_id, identifier = "scopus_id")
```

Here we will extract the abstract information from the result:

```
sc_info = abstract$content$`abstracts-retrieval-response`
substr(sc_info$coredata$`dc:description`, 1, 76)
```

```
[1] "© 2018, The Author(s). We present a study of multiple sclerosis segmentation"
```

Here we can extract information about the authors of the paper:

```
sc_df = purrr::map_df(sc_info$authors[[1]],
  as.data.frame, stringsAsFactors = FALSE, make.names = FALSE)
head(sc_df[, c("ce.given.name", "ce.initials", "X.auid")])
```

```
  ce.given.name ce.initials      X.auid
1       Olivier          O.  8431704700
2        Audrey          A. 57203861434
3       Michaël          M. 57199507814
4      Baptiste          B. 57197801981
5       Florent          F. 57203867656
6       Mathieu          M. 57203864793
```

This information is located within the author data.frame from the full_data as well. Note, however that the information from author_data was obtained by searching Scopus for an author, whereas the information from abstract_retrieval was obtained by searching for a specific paper. As we took the first entry from the Scopus identifier, we will subset the author data by entry_number 1 from the author data.frame to show the relevant info matches:

```
paper_author_info = jm$full_data$author
head(paper_author_info[paper_author_info$entry_number == 1, c("authid", "authname", "surname")])
```

```
       authid     authname   surname
1  8431704700 Commowick O. Commowick
2 57203861434    Istace A.    Istace
3 57199507814      Kain M.      Kain
4 57197801981   Laurent B.   Laurent
5 57203867656     Leray F.     Leray
6 57203864793     Simon M.     Simon
```

Thus, if we retrieve a single author's information, we can gather other author IDs from this directly. If we have a specific paper, we can retrieve author IDs from that paper information as well.

Object retrieval

Along with metadata, abstract, and other information about a paper, additional objects may be accessed, such as figures, videos, or supplemental material The object_retrieval function will return the objects associated with the identifier. The process_object_retrieval will convert the output into a tidy data.frame:

```
objects = object_retrieval("S1053811915002700", identifier = "pii", verbose = FALSE)
obj_df = process_object_retrieval(objects)
head(obj_df[, c("type", "url", "mime_type")])
```

```
             type
1 IMAGE-THUMBNAIL
2 IMAGE-THUMBNAIL
3 IMAGE-THUMBNAIL
4 IMAGE-THUMBNAIL
5 IMAGE-THUMBNAIL
6 IMAGE-THUMBNAIL
```

```
                                                                                 url
1 https://api.elsevier.com/content/object/eid/1-s2.0-S1053811915002700-gr1.sml?httpAccept=image/gif
2 https://api.elsevier.com/content/object/eid/1-s2.0-S1053811915002700-gr2.sml?httpAccept=image/gif
3 https://api.elsevier.com/content/object/eid/1-s2.0-S1053811915002700-gr3.sml?httpAccept=image/gif
4 https://api.elsevier.com/content/object/eid/1-s2.0-S1053811915002700-gr4.sml?httpAccept=image/gif
5 https://api.elsevier.com/content/object/eid/1-s2.0-S1053811915002700-gr5.sml?httpAccept=image/gif
6 https://api.elsevier.com/content/object/eid/1-s2.0-S1053811915002700-gr6.sml?httpAccept=image/gif
  mime_type
1 image/gif
2 image/gif
3 image/gif
4 image/gif
5 image/gif
6 image/gif
```

where the URL provides the link to download the object. Here we will subset the output to high-resolution images, and pass the URL for the first to the `download_object` function. This will return the content of the image, as well as download the image to disk (with path `outfile`):

```
obj_df = obj_df[ grepl("image/jpeg", obj_df$mime_type),]
obj_df = obj_df[ obj_df$type %in% "IMAGE-HIGH-RES",]
object = download_object(obj_df$url[1])
object$outfile
```

```
[1] "/var/folders/1s/wrtqcpxn685_zk570bnx9_rr0000gr/T//Rtmp43Omnj/file1fe21af69b3c.jpg"
```

One can then view the output using the system viewer using `utils::browseURL`. The `download_objects` function is a convenience wrapper for `lapply(url, download_object)` when passing in all the URLs from the output of `object_retrieval`.

Conclusion

The **rscopus** package provides an interface to the Scopus API through R. The package allows users to retrieve information about authors, individual articles, or institutions from the API. The information that can be obtained is limited by the scopes enabled by the API key through Elsevier and the associated organization. We have shown how to obtain an API key and specific use cases, which we believe covers the a number of the needs of the users.

More advanced usage requires additional information about the Scopus API and its querying specifications. The future direction of this package is to create helper functions to construct queries for more targeted views of the data. Additional APIs from Elsevier have functions for access, such as SciDirect with the `rscopus::sciencedirect_search` function, but the main focus of the package has been Scopus. Although the goal is to keep all functionality up to date with the number of APIs, the main focus will be consistency and stability with respect to Scopus. All the code and figures used to create this paper are located at `https://github.com/muschellij2/scopus` and the paper was created using the **rticles** package (Allaire et al., 2018).

Supplemental Material

Here is the simple parser `unique_title` to find the first 3 relevant words of the title after removing non-relevant words from a list in the **stopwords** package (Benoit et al., 2017):

```
unique_title = function(x) {
  ss = sapply(strsplit(x, split = " "),
              function(x) {
                x = x[ !tolower(x) %in% stopwords::stopwords()]
                x = x[ !x %in% c("-", "?", "--", 1:100)]
                paste(x[1:3], collapse = " ")
              })
  stopifnot(length(unique(ss)) == length(unique(x)))
  ss
}
```

Bibliography

J. Allaire, Y. Xie, R Foundation, H. Wickham, Journal of Statistical Software, R. Vaidyanathan, Association for Computing Machinery, C. Boettiger, Elsevier, K. Broman, K. Mueller, B. Quast, R. Pruim, B. Marwick, C. Wickham, O. Keyes, M. Yu, D. Emaasit, T. Onkelinx, A. Gasparini, M.-A. Desautels, D. Leutnant, MDPI, O. Öğreden, D. Hance, and D. Nüst. *rticles: Article Formats for R Markdown*, 2018. URL https://CRAN.R-project.org/package=rticles. R package version 0.6. [p13]

M. Aria and C. Cuccurullo. bibliometrix: An R-tool for comprehensive science mapping analysis. *Journal of Informetrics*, 11(4):959–975, 2017. URL https://doi.org/10.1016/j.joi.2017.08.007. [p1]

K. Benoit, D. Muhr, and K. Watanabe. *stopwords: Multilingual Stopword Lists*, 2017. URL https://CRAN.R-project.org/package=stopwords. R package version 0.9.0. [p13]

L. Egghe. Theory and practise of the g-index. *Scientometrics*, 69(1):131–152, 2006. [p6]

L. Henry and H. Wickham. *purrr: Functional Programming Tools*, 2018. URL https://CRAN.R-project.org/package=purrr. R package version 0.2.5. [p8]

J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National academy of Sciences*, 102(46):16569–16572, 2005. [p5]

J. Keirstead. *scholar: analyse citation data from Google Scholar*, 2016. URL http://github.com/jkeirstead/scholar. R package version 0.1.5. [p1]

J. Muschelli. *gcite: Google Citation Parser*, 2018. URL https://CRAN.R-project.org/package=gcite. R package version 0.9.2. [p1]

J. Ooms. The jsonlite package: A practical and consistent mapping between JSON data and R objects. *arXiv:1403.2805 [stat.CO]*, 2014. URL https://arxiv.org/abs/1403.2805. [p4]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL http://ggplot2.org. [p6]

H. Wickham. *httr: Tools for Working with URLs and HTTP*, 2017. URL https://CRAN.R-project.org/package=httr. R package version 1.3.1. [p4]

H. Wickham and L. Henry. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*, 2018. URL https://CRAN.R-project.org/package=tidyr. R package version 0.8.1. [p10]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. URL https://CRAN.R-project.org/package=dplyr. R package version 0.7.7. [p4]

H. Wickham et al. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. [p8]

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