Untitled

Clarke Iakovakis

May 29, 2019

Downloading R & R Studio

To download R, go to https://www.r-project.org/. Click on CRAN (Comprehensive R Archive Network) under Download, and scroll down to your country. Select the download link corresponding to the city that is geographically closest to you.

RStudio is a user interface for working with R. It is called an Integrated Development Environment (IDE) and acts as a sort of wrapper around the R language. You can use R without RStudio, but it's much more limiting. RStudio makes it easier to import datasets, create and write scripts, and has an autocomplete activated for functions and variables you've already assigned. RStudio makes using R much more effective, and is also free and open source.

Go to https://www.rstudio.com/products/RStudio/#Desktop to download the RStudio desktop software.

There are other IDEs such as Microsoft R Open and Notepad ++. Experiment with one that is right for you.

`rcrossref`

`crossref` is a package developed by Scott Chamberlain, Hao Zhu, Najko Jahn, Carl Boettiger, and Karthik Ram, part of the rOpenSci set of packages. It serves as an interface to the Crossref API.

Key links

- rcrossref documentation
- Crossref REST API documentation
- Crossref Metadata API JSON Format
- Tutorials
  - rOpenSci rcrossref tutorial
  - Paul Oldham: "Accessing the Scientific Literature with Crossref"
  - rcrossref vignette
- R cheat sheets
  - base R cheat sheet
  - purrr cheat sheet
  - data transformation cheat sheet
  - data import cheat sheet

Setting up `rcrossref`

First install and load `rcrossref` in R. Also install `usethis` in order to send your email to Crossref, as well as `tidyverse`, which is a package that contains many packages we'll use throughout.

```
install.packages("rcrossref")
install.packages("usethis")
install.packages("tidyverse")
install.packages("curl")
library(rcrossref)
library(usethis)
```

```
library(tidyverse)
library(curl)
```

As described in the documentation, the Crossref team encourages requests with appropriate contact information, and will forward you to a dedicated API cluster for improved performance when you share your email with them. Learn more here. To do so, first open your R environment using the `edit_r_environ()` function from the `usethis` package.

```
usethis::edit_r_environ()
```

A new window will open in R Studio. Type `crossref_email=name@example.com`, replacing the example with your own email address. Then press enter to create a new line, and leave it blank. Press Ctrl/Cmd + S to save the API key to your R environment and close the window. Your email address will now be shared with Crossref.

Getting publications from journals with `cr_journals`

`cr_journals()` takes either an ISSN or a general keyword query, and returns metadata for articles published in the journal, including DOI, title, volume, issue, pages, publisher, authors, etc. A full list of publications in Crossref is available on their website.

Getting journal details

Crossref is entirely dependant on publishers to supply the metadata. Some fields are required, while others are optional. You may therefore first be interested in what metadata publishers have submitted to Crossref for a given journal. By using `cr_journals` with `works = FALSE`, you can determine who publishes the journal, the total number of articles for the journal in Crossref, whether abstracts are included, if the full text of articles is deposited, if author ORCIDs are provided, and if the publisher supplies author affiliations, author ORCID iDs, article licensing data, funders for the article, article references, and a few other items.

First we will create a new vector `plosone_issn` with the ISSN for the journal *PLoS ONE*. We will then run `cr_journals()`, setting the ISSN equal to the `plosone_issn` we just created.

```
plosone_issn <- '1932-6203'
plosone_details <- rcrossref::cr_journals(issn = plosone_issn, works = FALSE) %>%
  purrr::pluck("data")
plosone_details
```

```
##      title                 publisher      issn last_status_check_time
## 1 PLoS ONE Public Library of Science 1932-6203             2019-03-24
##   deposits_abstracts_current deposits_orcids_current deposits
## 1                      FALSE                    TRUE     TRUE
##   deposits_affiliations_backfile deposits_update_policies_backfile
## 1                          FALSE                              TRUE
##   deposits_similarity_checking_backfile deposits_award_numbers_current
## 1                                  TRUE                             TRUE
##   deposits_resource_links_current deposits_articles
## 1                           FALSE              TRUE
##   deposits_affiliations_current deposits_funders_current
## 1                         FALSE                     TRUE
##   deposits_references_backfile deposits_abstracts_backfile
## 1                         TRUE                       FALSE
##   deposits_licenses_backfile deposits_award_numbers_backfile
```

```
## 1                       TRUE                            TRUE
##   deposits_open_references_backfile deposits_open_references_current
## 1                              TRUE                               TRUE
##   deposits_references_current deposits_resource_links_backfile
## 1                        TRUE                             FALSE
##   deposits_orcids_backfile deposits_funders_backfile
## 1                     TRUE                     TRUE
##   deposits_update_policies_current deposits_similarity_checking_current
## 1                             TRUE                                  TRUE
##   deposits_licenses_current affiliations_current
## 1                      TRUE                   0
##   similarity_checking_current funders_backfile licenses_backfile
## 1                   0.9999773       0.06110075         0.9907358
##   funders_current affiliations_backfile resource_links_backfile
## 1       0.6454721                    0                        0
##   orcids_backfile update_policies_current open_references_backfile
## 1      0.02560634                       1                        1
##   orcids_current similarity_checking_backfile references_backfile
## 1      0.8151801                    0.8933652           0.8933652
##   award_numbers_backfile update_policies_backfile licenses_current
## 1             0.05206973                0.9999417        0.9999773
##   award_numbers_current abstracts_backfile resource_links_current
## 1             0.5487785                  0                       0
##   abstracts_current open_references_current references_current total_dois
## 1                 0                        1          0.9999773     215525
##   current_dois backfile_dois
## 1        44005        171520
```

It actually comes back as a list of three items, two of which are unnecessary, and only one of which that contains the actual data. We use the `pluck()` function from the `purrr` package to pull that data only, which comes in a list element called "data". We will be using `pluck` throughout this tutorial; it's an easy way of indexing deeply and flexibly into lists to extract information.

The `purrr::pluck()` function is connected to `plosone_details` with something called a Pipe Operator `%>%`, which We will also be using throughout the tutorial. A pipe takes the output of one statement and immediately makes it the input of the next statement. It helps so that you don't have to write every intermediate, processing data to your R environment. You can think of it as "then" in natural language. So the above script first makes the API call with `cr_journals()`, then it applies `pluck()` to extract only the list element called "data", and returns it to the `plosone_details` value.

Looking at the data itsel by clicking on it in the R environment or calling `View(plosone_details)`, we see it includes one observation of 53 different variables, some of which I described above. I could not find the documentation for what these variables are anywhere on Crossref's webpage. We can either click on the data frame and browse it, or print any of these to the console, such as the publisher:

```
plosone_details$publisher
```

```
## [1] "Public Library of Science"
```

The total number of DOIs on file:

```
plosone_details$total_dois
```

```
## [1] 215525
```

Whether they deposit data on funders of research (a TRUE/FALSE value–called "logical" in R:

```
plosone_details$deposits_funders_current
```

```
## [1] TRUE
```

And so on. Call `names(plosone_details)` to print just the names of the 53 variables to the console:

```
names(plosone_details)
```

```
##  [1] "title"
##  [2] "publisher"
##  [3] "issn"
##  [4] "last_status_check_time"
##  [5] "deposits_abstracts_current"
##  [6] "deposits_orcids_current"
##  [7] "deposits"
##  [8] "deposits_affiliations_backfile"
##  [9] "deposits_update_policies_backfile"
## [10] "deposits_similarity_checking_backfile"
## [11] "deposits_award_numbers_current"
## [12] "deposits_resource_links_current"
## [13] "deposits_articles"
## [14] "deposits_affiliations_current"
## [15] "deposits_funders_current"
## [16] "deposits_references_backfile"
## [17] "deposits_abstracts_backfile"
## [18] "deposits_licenses_backfile"
## [19] "deposits_award_numbers_backfile"
## [20] "deposits_open_references_backfile"
## [21] "deposits_open_references_current"
## [22] "deposits_references_current"
## [23] "deposits_resource_links_backfile"
## [24] "deposits_orcids_backfile"
## [25] "deposits_funders_backfile"
## [26] "deposits_update_policies_current"
## [27] "deposits_similarity_checking_current"
## [28] "deposits_licenses_current"
## [29] "affiliations_current"
## [30] "similarity_checking_current"
## [31] "funders_backfile"
## [32] "licenses_backfile"
## [33] "funders_current"
## [34] "affiliations_backfile"
## [35] "resource_links_backfile"
## [36] "orcids_backfile"
## [37] "update_policies_current"
## [38] "open_references_backfile"
## [39] "orcids_current"
## [40] "similarity_checking_backfile"
## [41] "references_backfile"
## [42] "award_numbers_backfile"
## [43] "update_policies_backfile"
## [44] "licenses_current"
## [45] "award_numbers_current"
## [46] "abstracts_backfile"
## [47] "resource_links_current"
```

```
## [48] "abstracts_current"
## [49] "open_references_current"
## [50] "references_current"
## [51] "total_dois"
## [52] "current_dois"
## [53] "backfile_dois"
```

Getting journal publications by ISSN

To get metadata for the publications themselves rather than data about the journal, we will again use the `plosone_issn` value in the `issn =` argument to `cr_journals`, but we now set `works = TRUE` (you can use a `T` or the full word `TRUE`). The default number of articles returned is 20, but you can increase or decrease that with the `limit` argument. The max limit is 1000, but you can get more using the `cursor` argument (see below). As above, we use `pluck()` to return only the list element called "data", which is where the meat is.

```
plosone_publications <- rcrossref::cr_journals(issn = plosone_issn, works = T, limit = 25) %>%
  purrr::pluck("data")
plosone_publications
```

```
## # A tibble: 25 x 27
##    container.title created deposited published.online doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>            <chr> <chr>   <chr>
##  1 PLOS ONE        2019-0~ 2019-04-~ 2019-04-16       10.1~ 2019-0~ 1932~
##  2 PLoS ONE        2013-1~ 2018-10-~ 2013-12-03       10.1~ 2019-0~ 1932~
##  3 PLOS ONE        2015-0~ 2018-10-~ 2015-01-30       10.1~ 2019-0~ 1932~
##  4 PLoS ONE        2014-0~ 2018-10-~ 2014-09-30       10.1~ 2019-0~ 1932~
##  5 PLOS ONE        2018-0~ 2018-02-~ 2018-02-14       10.1~ 2019-0~ 1932~
##  6 PLoS ONE        2013-0~ 2017-06-~ 2013-06-21       10.1~ 2019-0~ 1932~
##  7 PLOS ONE        2015-0~ 2017-06-~ 2015-09-17       10.1~ 2019-0~ 1932~
##  8 PLoS ONE        2013-1~ 2018-10-~ 2013-10-25       10.1~ 2019-0~ 1932~
##  9 PLOS ONE        2015-1~ 2017-06-~ 2015-11-11       10.1~ 2019-0~ 1932~
## 10 PLOS ONE        2015-1~ 2018-10-~ 2015-12-11       10.1~ 2019-0~ 1932~
## # ... with 15 more rows, and 20 more variables: issue <chr>, issued <chr>,
## #   member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, subject <chr>,
## #   title <chr>, type <chr>, update.policy <chr>, url <chr>, volume <chr>,
## #   author <list>, link <list>, license <list>, reference <list>,
## #   funder <list>
```

You can also pass multiple ISSNs to `cr_journals`. Here we create 2 new values, `jama_issn` and `jah_issn`. These are ISSNs for the *Journal of American History* and *JAMA: The Journal of the American Medical Association*. We then pass them to `cr_journals` by passing them to the `c()` function, which will combine them (it's like CONCATENATE in Excel). We set `works` to `TRUE` so we'll get the publications metadata, and we set the `limit` to 50, so we'll get 50 publications per journal.

```
jama_issn <- '1538-3598'
jah_issn <- '0021-8723'
jah_jama_publications <- rcrossref::cr_journals(issn = c(jama_issn, jah_issn), works = T, limit = 50) %
  purrr::pluck("data")
jah_jama_publications
```

```
## # A tibble: 100 x 24
##    container.title created deposited published.print doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>           <chr> <chr>   <chr>
##  1 JAMA: The Jour~ 2003-0~ 2011-03-~ 1970-08-31      10.1~ 2019-0~ 0098~
```

```
##  2 JAMA: The Jour~ 2003-0~ 2011-08-~ 1984-10-12      10.1~ 2019-0~ 0098~
##  3 JAMA: The Jour~ 2003-0~ 2011-08-~ 1968-08-19      10.1~ 2019-0~ 0098~
##  4 JAMA: The Jour~ 2003-0~ 2007-02-~ 1990-05-16      10.1~ 2019-0~ 0098~
##  5 JAMA: The Jour~ 2003-0~ 2011-08-~ 1994-06-22      10.1~ 2019-0~ 0098~
##  6 JAMA: The Jour~ 2003-0~ 2011-08-~ 1987-05-15      10.1~ 2019-0~ 0098~
##  7 JAMA: The Jour~ 2003-0~ 2011-08-~ 1975-10-13      10.1~ 2019-0~ 0098~
##  8 JAMA: The Jour~ 2003-0~ 2011-08-~ 1996-08-21      10.1~ 2019-0~ 0098~
##  9 JAMA: The Jour~ 2003-0~ 2007-02-~ 1985-12-20      10.1~ 2019-0~ 0098~
## 10 JAMA: The Jour~ 2003-0~ 2011-04-~ 1976-12-13      10.1~ 2019-0~ 0098~
## # ... with 90 more rows, and 17 more variables: issue <chr>, issued <chr>,
## #   member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, volume <chr>, author <list>, link <list>,
## #   published.online <chr>, subject <chr>
```

A data frame of 100 observations of 24 variables is returned. This is a rich set of metadata for the articles in the given publications. The fields are detailed in the Crossref documentation, including the field name, type, description, and whether or not it's required. Some of these fields are title, DOI, DOI prefix identifer, ISSN, volume, issue, publisher, abstract (if provided), reference count (if provided–i.e., the number of references *in* the given article), link (if provided), subject (if provided), and other information. The number of citations *to* the article are not pulled, but these can be gathered separately with `cr_citation_count()` (see below).

Filtering rows and selecting columns with `dplyr`

You can use the `filter()` and `select()` functions from the `dplyr` package if you want to get subsets of this data after you have made the query. For instance, let's say you want only volume 99, issue 4 of the *Journal of American History*, and then want to keep only a few columns rather than all 24:

```
jah_99_4 <- jah_jama_publications %>%
  dplyr::filter(issn == jah_issn,
         volume == "99",
         issue == "4") %>%
  dplyr::select(title, doi, volume, issue, page, issued, issn)
jah_99_4
```

```
## # A tibble: 44 x 7
##    title                    doi        volume issue page   issued  issn
##    <chr>                    <chr>      <chr>  <chr> <chr>  <chr>   <chr>
##  1 Consultants in the Class~ 10.1093/j~ 99     4     1161-~ 2013-0~ 0021-~
##  2 Building People's Histor~ 10.1093/j~ 99     4     1176-~ 2013-0~ 0021-~
##  3 The Cultural Fieldwork I~ 10.1093/j~ 99     4     1189-~ 2013-0~ 0021-~
##  4 Homesickness: An America~ 10.1093/j~ 99     4     1199-~ 2013-0~ 0021-~
##  5 The Crucible of Consent:~ 10.1093/j~ 99     4     1198-~ 2013-0~ 0021-~
##  6 The Two Faces of America~ 10.1093/j~ 99     4     1201-~ 2013-0~ 0021-~
##  7 American Exceptionalisms~ 10.1093/j~ 99     4     1202-~ 2013-0~ 0021-~
##  8 Southern Stalemate: Five~ 10.1093/j~ 99     4     1311-~ 2013-0~ 0021-~
##  9 Black Internationalist F~ 10.1093/j~ 99     4     1309-~ 2013-0~ 0021-~
## 10 ACC Basketball: The Stor~ 10.1093/j~ 99     4     1312-~ 2013-0~ 0021-~
## # ... with 34 more rows
```

`filter()` will go through each row within the column specified and keep only those values matching the value you input. 'select() keeps only the variables you mention.

Note: be careful of filtering by ISSN. If a journal has multiple ISSNs they'll be combined in a single cell with a comma and the `filter()` will fail, as with JAMA. In this case it may be wiser to use `str_detect()`, as

described a couple code chunks down.

```
jah_jama_publications$issn[1]
```

```
## [1] "0098-7484,1538-3598"
```

And of course we can get a single article if we need, either by DOI:

```
jama_article <- jah_jama_publications %>%
  dplyr::filter(doi == "10.1001/jama.244.16.1799") %>%
  dplyr::select(title, doi, volume, issue, page, issued, issn)
jama_article
```

```
## # A tibble: 1 x 7
##   title                doi          volume issue page   issued  issn
##   <chr>                <chr>        <chr>  <chr> <chr>  <chr>   <chr>
## 1 The acute cardiac r~ 10.1001/jam~ 244    16    1799-~ 1980-1~ 0098-7484~
```

Or by title:

```
jah_article <- jah_jama_publications %>%
  dplyr::filter(stringr::str_detect(title, "Gridiron University")) %>%
  dplyr::select(title, doi, volume, issue, page, issued, issn)
jah_article
```

```
## # A tibble: 1 x 7
##   title                      doi        volume issue page   issued  issn
##   <chr>                      <chr>      <chr>  <chr> <chr>  <chr>   <chr>
## 1 The Rise of Gridiron Univ~ 10.1093/j~ 99     4     1277-~ 2013-0~ 0021-~
```

We use the `str_detect()` function from the `stringr` package, which is loaded as part of the `tidyverse`, in order to find a single term in the title, instead of requiring an exact match of the entire title.

Filtering the `cr_journals` query with the `filter` argument

Rather than pulling a large number of articles and then filtering after the fact with the `dplyr` functions, you can use the `filter` argument within `cr_journals` to specify some parameters as the query is executing. Note that this `filter` is completely different from the one we just used in `dplyr`. This filter is built into the Crossref API query. See the available filters by calling `rcrossref::filter_names()`, and details by calling `rcrossref::filter_details`. It's also in the API documentation.

Filtering by publication date with `from_pub_date` and `until_pub_date`

For example, you may only want to pull publications from a given year, or within a date range. Remember to increase the limit or use `cursor` if you need to. Also notice three things about the `filter` argument:

- The query parameter is in backticks (the key next to the 1 on the keyboard)
- The query itself is in single quotes
- The whole thing is wrapped in `c()`

```
jlsc_issn <- "2162-3309"
jlsc_publications_2017 <- rcrossref::cr_journals(issn = jlsc_issn, works = T, limit = 1000,
                                                 filter = c(from_pub_date='2017-08-01')) %>%
  purrr::pluck("data")
jlsc_publications_2017
```

```
## # A tibble: 47 x 21
```

```
##    container.title created deposited published.online doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>            <chr> <chr>   <chr>
##  1 Journal of Lib~ 2019-0~ 2019-04-~ 2019-04-11       10.7~ 2019-0~ 2162~
##  2 Journal of Lib~ 2018-0~ 2018-03-~ 2018             10.7~ 2019-0~ 2162~
##  3 Journal of Lib~ 2018-0~ 2018-05-~ 2018-05-08       10.7~ 2019-0~ 2162~
##  4 Journal of Lib~ 2018-0~ 2018-04-~ 2018-04-30       10.7~ 2019-0~ 2162~
##  5 Journal of Lib~ 2018-0~ 2018-05-~ 2018-05-03       10.7~ 2019-0~ 2162~
##  6 Journal of Lib~ 2018-0~ 2018-05-~ 2018-05-25       10.7~ 2019-0~ 2162~
##  7 Journal of Lib~ 2018-0~ 2018-06-~ 2018-06-12       10.7~ 2019-0~ 2162~
##  8 Journal of Lib~ 2018-0~ 2018-06-~ 2018-06-06       10.7~ 2019-0~ 2162~
##  9 Journal of Lib~ 2018-0~ 2018-07-~ 2018-07-25       10.7~ 2019-0~ 2162~
## 10 Journal of Lib~ 2018-0~ 2018-07-~ 2018-07-17       10.7~ 2019-0~ 2162~
## # ... with 37 more rows, and 14 more variables: issue <chr>, issued <chr>,
## #   member <chr>, prefix <chr>, publisher <chr>, reference.count <chr>,
## #   score <chr>, source <chr>, title <chr>, type <chr>, url <chr>,
## #   volume <chr>, author <list>, page <chr>
```

You can also use `until_pub_date`. See the table linked above for other dates, including online publication date, print publication date, posted date, and accepted date.

Filtering by license with `has_license`

You may be interested in licensing information for articles; for instance, gathering publications in a given journal that are licensed under Creative Commons. First run `cr_journals` with `works` set to `FALSE` in order to return journal details so you can check if the publisher even sends article licensing information to Crossref–it's not required. We will use PLOS ONE again as an example.

```
plosone_issn <- '1932-6203'
plosone_details <- rcrossref::cr_journals(issn = plosone_issn, works = FALSE) %>%
  purrr::pluck("data")
plosone_details$deposits_licenses_backfile
```

```
## [1] TRUE
```

```
plosone_details$deposits_licenses_current
```

```
## [1] TRUE
```

We can see that `deposits_licenses_backfile` is `TRUE` and `deposits_licenses_current` is also `TRUE`, meaning PLOS ONE does send licensing information and it is current. We can now rerun the query but set `works = TRUE`, and set the `has_license` to `TRUE`. This will therefore return only articles that have license information. We will set our `limit` to 25.

```
plosone_license <- rcrossref::cr_journals(issn = plosone_issn, works = T, limit = 25, filter = c(`has_li
  pluck("data")
plosone_license
```

```
## # A tibble: 25 x 27
##    container.title created deposited published.online doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>            <chr> <chr>   <chr>
##  1 PLOS ONE        2019-0~ 2019-04-~ 2019-04-16       10.1~ 2019-0~ 1932~
##  2 PLoS ONE        2013-1~ 2018-10-~ 2013-12-03       10.1~ 2019-0~ 1932~
##  3 PLOS ONE        2015-0~ 2018-10-~ 2015-01-30       10.1~ 2019-0~ 1932~
##  4 PLoS ONE        2014-0~ 2018-10-~ 2014-09-30       10.1~ 2019-0~ 1932~
##  5 PLOS ONE        2018-0~ 2018-02-~ 2018-02-14       10.1~ 2019-0~ 1932~
##  6 PLoS ONE        2013-0~ 2017-06-~ 2013-06-21       10.1~ 2019-0~ 1932~
```

```
##  7 PLOS ONE        2015-0~ 2017-06-~ 2015-09-17       10.1~ 2019-0~ 1932~
##  8 PLoS ONE        2013-1~ 2018-10-~ 2013-10-25       10.1~ 2019-0~ 1932~
##  9 PLOS ONE        2015-1~ 2017-06-~ 2015-11-11       10.1~ 2019-0~ 1932~
## 10 PLOS ONE        2015-1~ 2018-10-~ 2015-12-11       10.1~ 2019-0~ 1932~
## # ... with 15 more rows, and 20 more variables: issue <chr>, issued <chr>,
## #   member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, subject <chr>,
## #   title <chr>, type <chr>, update.policy <chr>, url <chr>, volume <chr>,
## #   author <list>, link <list>, license <list>, reference <list>,
## #   funder <list>
```

All articles in *PLoS ONE* are CC licensed, so we have no problem getting results. The license data comes in as a nested column. We can unnest it using `tidyr::unnest`, which will be discussed more below. For now, we will set `.drop = FALSE`, which will keep the other list columns (author, link, and funder).

```
plosone_license_unnested <- plosone_license %>%
  unnest(license, .drop = FALSE)
plosone_license_unnested
```

```
## # A tibble: 25 x 30
##    container.title created deposited published.online doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>            <chr> <chr>   <chr>
##  1 PLOS ONE        2019-0~ 2019-04-~ 2019-04-16       10.1~ 2019-0~ 1932~
##  2 PLoS ONE        2013-1~ 2018-10-~ 2013-12-03       10.1~ 2019-0~ 1932~
##  3 PLOS ONE        2015-0~ 2018-10-~ 2015-01-30       10.1~ 2019-0~ 1932~
##  4 PLoS ONE        2014-0~ 2018-10-~ 2014-09-30       10.1~ 2019-0~ 1932~
##  5 PLOS ONE        2018-0~ 2018-02-~ 2018-02-14       10.1~ 2019-0~ 1932~
##  6 PLoS ONE        2013-0~ 2017-06-~ 2013-06-21       10.1~ 2019-0~ 1932~
##  7 PLOS ONE        2015-0~ 2017-06-~ 2015-09-17       10.1~ 2019-0~ 1932~
##  8 PLoS ONE        2013-1~ 2018-10-~ 2013-10-25       10.1~ 2019-0~ 1932~
##  9 PLOS ONE        2015-1~ 2017-06-~ 2015-11-11       10.1~ 2019-0~ 1932~
## 10 PLOS ONE        2015-1~ 2018-10-~ 2015-12-11       10.1~ 2019-0~ 1932~
## # ... with 15 more rows, and 23 more variables: issue <chr>, issued <chr>,
## #   member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, subject <chr>,
## #   title <chr>, type <chr>, update.policy <chr>, url <chr>, volume <chr>,
## #   author <list>, link <list>, reference <list>, funder <list>,
## #   date <chr>, URL <chr>, delay.in.days <int>, content.version <chr>
```

This adds four columns all the way to the right: **date** (Date on which this license begins to take effect), **URL** (Link to a web page describing this license–in this case, Creative Commons), **delay in days** (Number of days between the publication date of the work and the start date of this license), and **content.version**, which specifies the version of the article the licensing data pertains to (VOR = Version of Record, AM = Accepted Manuscript, TDM = Text and Data Mining). Browsing the rows, we see most are CC BY 3.0 or 4.0.

Field queries

At the risk of confusing you further (but in the interests of providing more information), there is yet another way of making your query more precise, and that is to use a field query (`flq`) argument to `cr_journals()`. This allows you to specify additional variables, which are listed in the Crossref documentation and reproduced below. You *must* provide an ISSN–in other words, you can't run a field query for authors across all journals.

field_query_parameter	description
query.title	Query title and subtitle
query.container-title	Query container-title aka. publication name
query.author	Query author given and family names
query.editor	Query editor given and family names
query.chair	Query chair given and family names
query.translator	Query translator given and family names
query.contributor	Query author, editor, chair and translator given and family names
query.bibliographic	Query bibliographic information, useful for citation look up. Includes titles, authors, ISSNs and
query.affiliation	Query contributor affiliations

Field query by title

Here, we get all publications from the Journal of Librarianship and Scholarly Communication with the term "open access" in the title.

```
jlsc_publications_oa <- rcrossref::cr_journals(issn = jlsc_issn, works = T, limit = 1000,
                                               flq = c(`query.title` = 'open access')) %>%
  purrr::pluck("data")
jlsc_publications_oa
```

```
## # A tibble: 48 x 21
##    container.title created deposited published.online doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>            <chr> <chr>   <chr>
##  1 Journal of Lib~ 2015-0~ 2015-09-~ 2015             10.7~ 2019-0~ 2162~
##  2 Journal of Lib~ 2012-0~ 2015-09-~ 2012             10.7~ 2019-0~ 2162~
##  3 Journal of Lib~ 2013-0~ 2015-09-~ 2013             10.7~ 2019-0~ 2162~
##  4 Journal of Lib~ 2013-1~ 2015-09-~ 2013             10.7~ 2019-0~ 2162~
##  5 Journal of Lib~ 2016-0~ 2016-02-~ 2016             10.7~ 2019-0~ 2162~
##  6 Journal of Lib~ 2017-0~ 2017-04-~ 2017             10.7~ 2019-0~ 2162~
##  7 Journal of Lib~ 2012-0~ 2015-09-~ 2012             10.7~ 2019-0~ 2162~
##  8 Journal of Lib~ 2012-0~ 2015-09-~ 2012             10.7~ 2019-0~ 2162~
##  9 Journal of Lib~ 2012-0~ 2015-09-~ 2012             10.7~ 2019-0~ 2162~
## 10 Journal of Lib~ 2012-0~ 2015-09-~ 2012             10.7~ 2019-0~ 2162~
## # ... with 38 more rows, and 14 more variables: issue <chr>, issued <chr>,
## #   member <chr>, prefix <chr>, publisher <chr>, reference.count <chr>,
## #   score <chr>, source <chr>, title <chr>, type <chr>, url <chr>,
## #   volume <chr>, author <list>, page <chr>
```

Field query by author, contributor, or editor

The `flq` argument can also be used for authors, contributors, or editors. Here we search the same journal for authors with the name Salo (looking for all articles written by Dorothea Salo).

```
jlsc_publications_auth <- rcrossref::cr_journals(issn = jlsc_issn, works = T, limit = 1000,
                                               flq = c(`query.author` = 'salo')) %>%
  purrr::pluck("data")
jlsc_publications_auth
```

```
## # A tibble: 2 x 20
##   container.title created deposited published.online doi   indexed issn
##   <chr>           <chr>   <chr>     <chr>            <chr> <chr>   <chr>
## 1 Journal of Lib~ 2013-0~ 2015-09-~ 2013             10.7~ 2019-0~ 2162~
```

```
## 2 Journal of Lib~ 2013-1~ 2015-09-~ 2013             10.7~ 2019-0~ 2162~
## # ... with 13 more variables: issue <chr>, issued <chr>, member <chr>,
## #   prefix <chr>, publisher <chr>, reference.count <chr>, score <chr>,
## #   source <chr>, title <chr>, type <chr>, url <chr>, volume <chr>,
## #   author <list>
```

Unnesting list columns

Authors, links, licenses, funders, and some other values can appear in nested lists when you call `cr_journals` because there can be, and often are, multiple of each of these items per article. Our `jah_jama_publications` data frame has nested lists for **author** and **link**. You can check the data classes on all variables by running `typeof()` across all columns using the `map_chr()` function from `purrr`:

```
purrr::map_chr(jah_jama_publications, typeof)
```

```
##  container.title          created        deposited  published.print
##      "character"      "character"      "character"      "character"
##              doi          indexed             issn            issue
##      "character"      "character"      "character"      "character"
##           issued           member             page           prefix
##      "character"      "character"      "character"      "character"
##        publisher  reference.count            score           source
##      "character"      "character"      "character"      "character"
##            title             type              url           volume
##      "character"      "character"      "character"      "character"
##           author             link published.online          subject
##           "list"           "list"      "character"      "character"
```

These can normally be easily unnested using the `unnest()` function from the `tidyr` package. However it throws an error if there are NULL values (when the publisher hasn't provided that data to Crossref). It looks like the `tidyr` developer is currently working to allow NULL while unnesting but until then, we need a solution if we want to unnest the lists.

This is a bit complicated, but you can use `map_lgl` to first remove (`filter()` out) the `NULL` author columns, then `unnest` the authors, then bind back the `NULL` author columns, and finally deselecting the extra `author` and `link` columns as these are no longer in the transformed, unnested data.

```
jah_jama_publications_auth <- jah_jama_publications %>%
  dplyr::filter(!purrr::map_lgl(author, is.null)) %>%
  tidyr::unnest(author, .drop = TRUE) %>%
  dplyr::bind_rows(jah_jama_publications %>%
                     dplyr::filter(map_lgl(author, is.null)) %>%
                     dplyr::select(-author, -link))
jah_jama_publications_auth
```

```
## # A tibble: 105 x 25
##    container.title created deposited published.print doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>           <chr> <chr>   <chr>
##  1 JAMA: The Jour~ 2003-0~ 2011-03-~ 1970-08-31      10.1~ 2019-0~ 0098~
##  2 JAMA: The Jour~ 2003-0~ 2011-08-~ 1984-10-12      10.1~ 2019-0~ 0098~
##  3 JAMA: The Jour~ 2003-0~ 2011-08-~ 1968-08-19      10.1~ 2019-0~ 0098~
##  4 JAMA: The Jour~ 2003-0~ 2007-02-~ 1990-05-16      10.1~ 2019-0~ 0098~
##  5 JAMA: The Jour~ 2003-0~ 2011-08-~ 1994-06-22      10.1~ 2019-0~ 0098~
##  6 JAMA: The Jour~ 2003-0~ 2011-08-~ 1987-05-15      10.1~ 2019-0~ 0098~
##  7 JAMA: The Jour~ 2003-0~ 2011-08-~ 1975-10-13      10.1~ 2019-0~ 0098~
##  8 JAMA: The Jour~ 2003-0~ 2011-04-~ 1976-12-13      10.1~ 2019-0~ 0098~
```

```
##  9 JAMA: The Jour~ 2003-0~ 2011-05-~ 1981-07-03      10.1~ 2019-0~ 0098~
## 10 JAMA: The Jour~ 2003-0~ 2011-08-~ 1991-09-18      10.1~ 2019-0~ 0098~
## # ... with 95 more rows, and 18 more variables: issue <chr>, issued <chr>,
## #   member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, volume <chr>, published.online <chr>,
## #   subject <chr>, given <chr>, family <chr>, sequence <chr>
```

When we `unnest()` a nested list, we will see an additional row added for each variable with multiple authors. We will also see three new columns added: **given** (first name), **family** (last name), and **sequence** (first or additional). All other data for these rows will be duplicated except for those values.

We now have 105 observations of 25 variables, rather than 100 observations of 24 variables. So the rows with duplicated DOIs are the ones with multiple co-authors. The titles of these are

```
jah_jama_publications_auth %>%
  dplyr::filter(duplicated(doi)) %>%
  dplyr::select(title) %>%
  dplyr::distinct()
```

```
## # A tibble: 3 x 1
##   title
##   <chr>
## 1 Building People's Histories: Graduate Student Pedagogy, Undergraduate Ed~
## 2 The Cultural Fieldwork Initiative: Collaboration for Better Education
## 3 American Enlightenments: Continuity and Renewal
```

Out of that set of 100 articles, only 3 were co-authored.

We can do the same with **link**, just replace "author":

```
jah_jama_publications_link <- jah_jama_publications %>%
  dplyr::filter(!purrr::map_lgl(link, is.null)) %>%
  tidyr::unnest(link, .drop = FALSE) %>%
  dplyr::bind_rows(jah_jama_publications %>%
                     dplyr::filter(map_lgl(link, is.null)) %>%
                     dplyr::select(-author, -link))
jah_jama_publications_link
```

```
## # A tibble: 100 x 27
##    container.title created deposited published.print doi   indexed issn
##    <chr>           <chr>   <chr>     <chr>           <chr> <chr>   <chr>
##  1 The Journal of~ 2006-0~ 2017-08-~ 1973-09         10.2~ 2019-0~ 0021~
##  2 The Journal of~ 2006-0~ 2017-08-~ 1969-03         10.2~ 2019-0~ 0021~
##  3 Journal of Ame~ 2013-0~ 2017-08-~ 2013-09-01      10.1~ 2019-0~ 0021~
##  4 Journal of Ame~ 2014-0~ 2017-08-~ 2014-03-01      10.1~ 2019-0~ 0021~
##  5 The Journal of~ 2006-0~ 2017-08-~ 1997-06         10.2~ 2019-0~ 0021~
##  6 The Journal of~ 2008-0~ 2017-08-~ 2002-12         10.2~ 2019-0~ 0021~
##  7 Journal of Ame~ 2013-0~ 2017-08-~ 2013-03-01      10.1~ 2019-0~ 0021~
##  8 Journal of Ame~ 2013-0~ 2017-08-~ 2013-03-01      10.1~ 2019-0~ 0021~
##  9 Journal of Ame~ 2013-0~ 2017-08-~ 2013-03-01      10.1~ 2019-0~ 0021~
## 10 Journal of Ame~ 2013-0~ 2017-08-~ 2013-03-01      10.1~ 2019-0~ 0021~
## # ... with 90 more rows, and 20 more variables: issue <chr>, issued <chr>,
## #   member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, volume <chr>, author <list>,
## #   published.online <chr>, subject <chr>, URL <chr>, content.type <chr>,
```

```
## #   content.version <chr>, intended.application <chr>
```

It is important to point out that there is an argument to `unnest()` called `.drop`, which is set to `TRUE` by default, meaning all additional list columns in the data frame will be removed automatically. So in this case, when we unnest **author, link** is dropped as well. If you want to keep it, just set it to `FALSE` Note, however, that it will not unnest it unless or until you call `unnest()` on it.

Getting more than 1000 results with the `cursor` argument to `cr_journals`

If our result will have more than 1000 results, we have to use the `cursor` argument. Here we will look at the journal *Philosophical Transactions*, the longest running scientific journal in the world. We first run `cr_journals` with `works` set to `FALSE` in order to get the journal details, specifically to find out how many articles from this journal are indexed in Crossref.

```
philo_issn <- '2053-9223'
philo_details <- rcrossref::cr_journals(philo_issn, works = FALSE) %>%
  pluck("data")
philo_details$total_dois
```

```
## [1] 8534
```

Because there are 8,534 articles, we need to pass the `cursor` argument to `cr_journals`, which is called "deep paging." As described by Paul Oldham:

> "the CrossRef API also permits deep paging (fetching results from multiple pages). We can retrieve all the results by setting the cursor to the wildcard * and the cursor_max to the total results. Note that the cursor and the cursor_max arguments must appear together for this to work."

See more in the API documentation

Here I will set the limit to 1,500 rather than collect the entire 8,534.

```
philo_articles <- rcrossref::cr_journals(philo_issn, works = TRUE, cursor = "*", cursor_max = 1500) %>%
  pluck("data")
```

Writing publications to disk

We will use the `write_csv()` function from the `readr` package to write our data to disk. This package was loaded when you called `library(tidyverse)` at the beginning of the session.

First we're going to create a vector that represents file path to our desktop by grabbing our username and using the `file.path()` function. If you don't want to write all these files to your desktop, then you'll need to create a different file path to assign to `my_filepath`. This won't work on a Mac.

```
my_environment <- Sys.getenv("USERNAME")
my_filepath <- file.path("C:/Users", my_environment, "Desktop")
```

Unfortunately, you cannot simply write the `jah_jama_publications` data frame to a CSV, due to the nested lists. It will throw an error: `"Error in stream_delim_(df, path, ...) : Don't know how to handle vector of type list."`

You have a few choices here:

1. You can unnest one of the columns and leave `.drop` set to `TRUE`. This will add rows for all the values in the nested lists, and drop the additional nested lists.

```
jah_jama_publications_auth <- jah_jama_publications %>%
  dplyr::filter(!purrr::map_lgl(author, is.null)) %>%
  tidyr::unnest(author, .drop = TRUE) %>%
  dplyr::bind_rows(jah_jama_publications %>%
                     dplyr::filter(map_lgl(author, is.null)) %>%
                     dplyr::select(-author, -link))
readr::write_csv(jah_jama_publications_auth, file.path(my_filepath, "jah_jama_publications_auth.csv"))
```

2. You can drop the nested lists altogether using a combination of `select_if()` from `dplyr` and `negate()` from `purrr` to drop all lists in the data frame. This only works if you don't need author or link.

```
jah_jama_publications_short <- jah_jama_publications %>%
  dplyr::select_if(purrr::negate(is.list))
readr::write_csv(jah_jama_publications_author, file.path(my_filepath, "jah_jama_publications_short.csv"
```

We go from 24 to 22 variables because it dropped **author** and **list**.

3. You can use `mutate()` from `dplyr` to coerce the list columns into character vectors:

```
jah_jama_publications_mutated <- jah_jama_publications %>%
  dplyr::mutate(author = as.character(author)) %>%
  dplyr::mutate(link = as.character(link))
readr::write_csv(jah_jama_publications_mutated, file.path(my_filepath, "jah_jama_publications_mutated.cs
```

Using `cr_works` to get data on articles

This function allows you to search by DOI or a general query in order to return the Crossref metadata.

It is important to note, as Crossref does in the documentation:

> Crossref does not use "works" in the FRBR sense of the word. In Crossref parlance, a "work" is just a thing identified by a DOI. In practice, Crossref DOIs are used as citation identifiers. So, in FRBR terms, this means, that a Crossref DOI tends to refer to one *expression* which might include multiple *manifestations*. So, for example, the ePub, HTML and PDF version of an article will share a Crossref DOI because the differences between them should not effect the interpretation or crediting of the content. In short, they can be cited interchangeably. The same is true of the "accepted manuscript" and the "version-of-record" of that accepted manuscript.

Getting works from a typed citation in a Word document/text file

This can be helpful if you have a bibliography in a Word document or text file that you want to get into a reference management tool like Zotero. For instance, you may have written the citations in APA style and need to change to Chicago, but don't want to rekey it all out. Or perhaps you jotted down your citations hastily and left out volume, issue, or page numbers, and you need a nice, fully-formatted citation.

If each citation is on its own line in your document's bibliography, then you can probably paste the whole bibliography into an Excel spreadsheet. If it goes as planned, each citation will be in its own cell. You can then save it to a CSV file, which can then be read into R.

I mocked one up and saved it to my GitHub, so we'll connect to it by passing the URL on to `url()` and reading it as a CSV with `read_csv`.

```
my_references <- readr::read_csv(url("https://raw.githubusercontent.com/ciakovx/rorcid.workshop/master/
```

```
## Parsed with column specification:
```

```
## cols(
##   reference = col_character()
## )
```

```
my_references
```

```
## # A tibble: 8 x 1
##   reference
##   <chr>
## 1 Frosio, G. F. (2014). Open Access Publishing: A Literature Review. SSRN ~
## 2 "Laakso, M., & Bj\xf6rk, B.-C. (2016). Hybrid open access\x97A longitudi~
## 3 "Laakso, M., Welling, P., Bukvova, H., Nyman, L., Bj\xf6rk, B.-C., & Hed~
## 4 Paulus, F. M., Cruz, N., & Krach, S. (2018). The Impact Factor Fallacy. ~
## 5 Science, Digital; Hook, Daniel; Hahnel, Mark; Calvert, Ian (2019): The A~
## 6 "Shotton, D. (2018). Funders should mandate open citations. Nature, 553(~
## 7 Suber, P. (2012). Open access. Cambridge, Mass: MIT Press.
## 8 Tennant, J. P., Waldner, F., Jacques, D. C., Masuzzo, P., Collister, L. ~
```

As you can see, these are just raw citations, not divided into variables by their metadata elements (that is, with title in one column, author in another, etc.). But, we can now run a query to get precisely that from Crossref using `cr_works`. Because `cr_works` is not vectorized, we will need to build a loop using `map()` from the `purrr` package. This is the equivalent of copy/pasting the whole reference into the Crossref search engine. The loop will `print()` the citation before searching for it so we can keep track of where it is. We set the `limit` to 5 because if Crossref didn't find it in the first 5 results, it's not likely to be there at all.

```
my_references_works_list <- purrr::map(
  my_references$reference,
  function(x) {
    print(x)
    my_works <- rcrossref::cr_works(query = x, limit = 5) %>%
      purrr::pluck("data")
  })
```

```
## [1] "Frosio, G. F. (2014). Open Access Publishing: A Literature Review. SSRN Electronic Journal. http
## [1] "Laakso, M., & Bj<f6>rk, B.-C. (2016). Hybrid open access<U+0097>A longitudinal study. Journal o
## [1] "Laakso, M., Welling, P., Bukvova, H., Nyman, L., Bj<f6>rk, B.-C., & Hedlund, T. (2011). The Dev
## [1] "Paulus, F. M., Cruz, N., & Krach, S. (2018). The Impact Factor Fallacy. Frontiers in Psychology
## [1] "Science, Digital; Hook, Daniel; Hahnel, Mark; Calvert, Ian (2019): The Ascent of Open Access. f
## [1] "Shotton, D. (2018). Funders should mandate open citations. Nature, 553(7687), 129<U+0096>129. h
## [1] "Suber, P. (2012). Open access. Cambridge, Mass: MIT Press."
## [1] "Tennant, J. P., Waldner, F., Jacques, D. C., Masuzzo, P., Collister, L. B., & Hartgerink, C. H.
```

The Crossref API assigns a score to each item returned within each query, giving a measure of the API's confidence in the match. The item with the highest score is returned first in the datasets. We can return the first result in each item in the `my_references_works_list` by using `map_dfr()`, which is like `map()` except it returns the results into a data frame rather than a list:

```
my_references_works_df <- my_references_works_list %>%
  purrr::map_dfr(., function(x) {
    x[1, ]
  })
my_references_works_df
```

```
## # A tibble: 8 x 36
##   container.title created deposited doi   indexed issn  issued member
##   <chr>           <chr>   <chr>     <chr> <chr>   <chr> <chr>  <chr>
## 1 SSRN Electroni~ 2015-1~ 2015-12-~ 10.2~ 2019-0~ 1556~ 2014   78
```

```
## 2 Journal of Inf~ 2016-0~ 2018-09-~ 10.1~ 2019-0~ 1751~ 2016-~ 78
## 3 PLoS ONE        2011-0~ 2018-10-~ 10.1~ 2019-0~ 1932~ 2011-~ 340
## 4 Frontiers in P~ 2018-0~ 2018-08-~ 10.3~ 2019-0~ 1664~ 2018-~ 1965
## 5 <NA>            2017-0~ 2017-03-~ 10.7~ 2019-0~ <NA>  <NA>   4443
## 6 Nature          2018-0~ 2018-01-~ 10.1~ 2019-0~ 0028~ 2018-~ 297
## 7 Leonardo        2013-0~ 2016-12-~ 10.1~ 2019-0~ 0024~ 2013-~ 281
## 8 F1000Research   2016-0~ 2018-12-~ 10.1~ 2019-0~ 2046~ 2016-~ 2560
## # ... with 28 more variables: prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, author <list>, alternative.id <chr>,
## #   published.print <chr>, issue <chr>, page <chr>, update.policy <chr>,
## #   volume <chr>, assertion <list>, link <list>, license <list>,
## #   archive_ <chr>, na. <chr>, na..1 <chr>, published.online <chr>,
## #   abstract <chr>, reference <list>, subject <chr>, archive <chr>,
## #   isbn <chr>, funder <list>
```

Now we can print the titles to see how well they match with the titles of the works we requested:

```
my_references_works_df$title
```

```
## [1] "Open Access Publishing: A Literature Review"
## [2] "Hybrid open access-A longitudinal study"
## [3] "The Development of Open Access Journal Publishing from 1993 to 2009"
## [4] "The Impact Factor Fallacy"
## [5] "Table S1: Digital copies of visualizations, alignments, and phylogenetic trees deposited at Figs
## [6] "Funders should mandate open citations"
## [7] "Open Access by Peter Suber. MIT Press, Cambridge, MA, U.S.A., 2012. 230 pp. ISBN: 978-0- 262-51
## [8] "The academic, economic and societal impacts of Open Access: an evidence-based review"
```

Not bad! Looks like we got 6 out of 8, with problems on number 5 and 7.

Let's deal with 5 first. This was the result for "The Ascent of Open Access", which was a report by Digital Science posted to figshare, didn't come back. Even though this report does have a DOI (https://doi.org/10.6084/m9.figshare.7618751.v2) assigned via figshare, the `cr_works()` function searches only for Crossref DOIs. We should check to see if it came back in any of the 5 items we pulled. We do this by calling `pluck()` on the titles of the fifth item in the list:

```
my_references_works_list %>%
  purrr::pluck(5, "title")
```

```
## [1] "Table S1: Digital copies of visualizations, alignments, and phylogenetic trees deposited at Figs
## [2] "Figure 4 from: Miko I, Ernst A, Deans A (2013) Morphology and function of the ovipositor mechani
## [3] "Figure 1 from: Miko I, Ernst A, Deans A (2013) Morphology and function of the ovipositor mechani
## [4] "Figure 6 from: Miko I, Popovici O, Seltmann K, Deans A (2014) The maxillo-labial complex of Spar
## [5] "Figure 3 from: Miko I, Ernst A, Deans A (2013) Morphology and function of the ovipositor mechani
```

Nope, unfortunately none of these are "The Ascent of Open Access", so we're out of luck. We can just throw this row out entirely using `slice()` from `dplyr`. We'll overwrite our existing `my_references_works_df` because we have no future use for it in this R session.

```
my_references_works_df <- my_references_works_df %>%
  dplyr::slice(-5)
```

For row 7, it's giving us the full citation for Peter Suber's book when we asked for the title only, so something is fishy.

When we look at it more closely (we can call `View(my_references_works_df)`), we see the author of this item is not Peter Suber, but Rob Harle, and, checking the **type** column, it's a journal article, not a book. This is a book review published in the journal *Leonardo*, not Peter Suber's book. So let's go back to

`my_references_works_list` and pull data from all 5 items that came back with the API call and see if Suber's book is in there somewhere:

```
suber <- my_references_works_list %>%
  purrr::pluck(7)
suber
```

```
## # A tibble: 5 x 28
##   alternative.id container.title created deposited published.print doi
##   <chr>          <chr>           <chr>   <chr>     <chr>           <chr>
## 1 10.1162/LEON_~ Leonardo        2013-0~ 2016-12-~ 2013-04         10.1~
## 2 <NA>           <NA>            2019-0~ 2019-01-~ 2012            10.7~
## 3 10.1108/14684~ Online Informa~ 2014-1~ 2017-02-~ 2013-02-15      10.1~
## 4 <NA>           Open Access     2014-0~ 2018-10-~ 2006            10.1~
## 5 <NA>           BMJ             2012-0~ 2016-12-~ 2012-08-08      10.1~
## # ... with 22 more variables: indexed <chr>, issn <chr>, issue <chr>,
## #   issued <chr>, member <chr>, page <chr>, prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, subject <chr>,
## #   title <chr>, type <chr>, url <chr>, volume <chr>, author <list>,
## #   link <list>, reference <list>, isbn <chr>, license <list>,
## #   published.online <chr>
```

It looks like it is the second item, confirming by seeing the **author** is Peter Suber, the **publisher** is MIT Press, the **type** is book, and the **ISBN** is "9780262301732".

We do the following to correct it:

- use `filter()` with the isbn to assign the correct row from `suber` to a variable called `suber_correct`
- remove the incorrect row with `slice` (double checking that it is the 6th row)
- use `bind_rows()` to add the correct one to our `my_references_works_df` data frame. We can just overwrite the existing `my_references_works_df` again

```
suber_correct <- suber %>%
  dplyr::filter(isbn == "9780262301732")
my_references_works_df <- my_references_works_df %>%
  dplyr::slice(-6) %>%
  bind_rows(suber_correct)
```

Save to BibTeX file

We can now save these to a BibTeX file that can be read into a reference management software. We do this first with `rcrossref::cr_cn()` to format in BibTeX, then with `write_lines()` from the `readr` package.

```
my_references_dois <- my_references_works_df$doi
my_references_bibtex <- rcrossref::cr_cn(my_references_dois, format = "bibtex") %>%
  purrr::map_chr(., purrr::pluck, 1)
readr::write_lines(my_references_bibtex, file.path(my_filepath, "my_references_bibtex.bib"))
```

We again use `map_chr()` to scan each list item, `pluck()` out the first item and return a character. This gives us the full BibTeX reference for all 8 references.

If you are using Zotero, go to **File > Import**, then select "A file" and navigate to this .bib file we just created. Congratulations: you now have reference files!

We'll be covering more on `cr_cn()` below.

Running general queries on `cr_works()`

You can also use `cr_works()` to run a query based on very simple text keywords. For example, you can run `oa_works <- rcrossref::cr_works(query = "open+access")`. Paul Oldham gives a great example of this, but does make the comment:

> CrossRef is not a text based search engine and the ability to conduct text based searches is presently crude. Furthermore, we can only search a very limited number of fields and this will inevitably result in lower returns than commercial databases such as Web of Science (where abstracts and author keywords are available).

Unfortunately there is no boolean AND for Crossref queries (see https://github.com/CrossRef/rest-api-doc/issues/135 and https://twitter.com/CrossrefSupport/status/1073601263659610113). However, as discussed above, the Crossref API assigns a score to each item returned giving a measure of the API's confidence in the match, and if you connect words using `+` the Crossref API will give items with those terms a higher score.

Specifying field queries to `cr_works()` with `flq`

As with `cr_journals`, you can use `flq` to pass field queries on to `cr_works()`, such as author.

Here we search for the book *Open Access* by Peter Suber by doing a general keyword search for "open access" and an author search for "suber":

```
suber_oa <- cr_works(query = 'open+access', flq = c(`query.author` = 'suber')) %>%
  pluck("data")
suber_oa
```

```
## # A tibble: 16 x 29
##    container.title created deposited published.print doi   indexed isbn
##    <chr>           <chr>   <chr>     <chr>           <chr> <chr>   <chr>
##  1 Open Access     2014-0~ 2018-10-~ 2006            10.1~ 2019-0~ 9781~
##  2 <NA>            2019-0~ 2019-01-~ 2012            10.7~ 2019-0~ 9780~
##  3 The Journal of~ 2008-1~ 2019-04-~ <NA>            10.3~ 2019-0~ <NA>
##  4 The Journal of~ 2009-0~ 2019-04-~ <NA>            10.3~ 2019-0~ <NA>
##  5 <NA>            2019-0~ 2019-03-~ 2016            10.7~ 2019-0~ 9780~
##  6 Nature Geoscie~ 2009-0~ 2019-04-~ 2009-03         10.1~ 2019-0~ <NA>
##  7 Nature          2003-1~ 2018-02-~ 2003-11         10.1~ 2019-0~ <NA>
##  8 Logos           2011-0~ 2018-12-~ 2010-03-01      10.1~ 2019-0~ <NA>
##  9 Syllecta Class~ 2015-0~ 2015-04-~ 2005            10.1~ 2019-0~ <NA>
## 10 BMJ             2012-0~ 2016-12-~ 2012-08-08      10.1~ 2019-0~ <NA>
## 11 DESIDOC Journa~ 2014-0~ 2014-09-~ 2008-01-01      10.1~ 2019-0~ <NA>
## 12 DESIDOC Journa~ 2014-0~ 2014-09-~ 2008-01-01      10.1~ 2019-0~ <NA>
## 13 BMJ             2005-0~ 2018-02-~ 2005-05-14      10.1~ 2019-0~ <NA>
## 14 Ebooks in Educ~ 2014-1~ 2017-02-~ 2014-11-28      10.5~ 2019-0~ 9781~
## 15 College & Rese~ 2017-0~ 2017-06-~ <NA>            10.5~ 2019-0~ <NA>
## 16 Science         2007-0~ 2016-12-~ 2006-09-15      10.1~ 2019-0~ <NA>
## # ... with 22 more variables: issued <chr>, member <chr>, page <chr>,
## #   prefix <chr>, publisher <chr>, reference.count <chr>, score <chr>,
## #   source <chr>, title <chr>, type <chr>, url <chr>, author <list>,
## #   link <list>, license <list>, published.online <chr>, issn <chr>,
## #   issue <chr>, subject <chr>, volume <chr>, subtitle <chr>,
## #   alternative.id <chr>, reference <list>
```

Dr. Suber has written lots of materials that includes the term "open access." We can use the `filter()` function from `dplyr` to look only at books, from the **type** column:

```
suber_oa_books <- suber_oa %>%
  filter(type == "book")
suber_oa_books
```

```
## # A tibble: 2 x 29
##   container.title created deposited published.print doi   indexed isbn
##   <chr>           <chr>   <chr>     <chr>           <chr> <chr>   <chr>
## 1 <NA>            2019-0~ 2019-01-~ 2012            10.7~ 2019-0~ 9780~
## 2 <NA>            2019-0~ 2019-03-~ 2016            10.7~ 2019-0~ 9780~
## # ... with 22 more variables: issued <chr>, member <chr>, page <chr>,
## #   prefix <chr>, publisher <chr>, reference.count <chr>, score <chr>,
## #   source <chr>, title <chr>, type <chr>, url <chr>, author <list>,
## #   link <list>, license <list>, published.online <chr>, issn <chr>,
## #   issue <chr>, subject <chr>, volume <chr>, subtitle <chr>,
## #   alternative.id <chr>, reference <list>
```

One is the book from MIT Press that we're looking for; the other is *Knowledge Unbound*, which is a collection of his writings.

We could be more specific from the outset by adding bibliographic information in `query.bibliographic`, such as ISBN (or ISSN, if it's a journal):

```
suber_isbn <- cr_works(flq = c(`query.author` = 'suber',
                               `query.bibliographic` = '9780262301732')) %>%
  pluck("data")
suber_isbn
```

```
## # A tibble: 1 x 17
##   created deposited published.print doi   indexed isbn  issued member
##   <chr>   <chr>     <chr>           <chr> <chr>   <chr> <chr>  <chr>
## 1 2019-0~ 2019-01-~ 2012            10.7~ 2019-0~ 9780~ 2012   281
## # ... with 9 more variables: prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, author <list>
```

You can combine the `filter` argument with `flq` to return only items of **type** `book` published in 2012.

```
suber_filter_flq <- cr_works(filter = c(`type` = 'book'),
                             flq = c(`query.title` = 'open+access',
                                     `query.author` = 'suber',
                                     `query.bibliographic` = '2012')) %>%
  pluck("data")
suber_filter_flq
```

```
## # A tibble: 1 x 17
##   created deposited published.print doi   indexed isbn  issued member
##   <chr>   <chr>     <chr>           <chr> <chr>   <chr> <chr>  <chr>
## 1 2019-0~ 2019-01-~ 2012            10.7~ 2019-0~ 9780~ 2012   281
## # ... with 9 more variables: prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, author <list>
```

Querying Crossref works by DOI

The process is much easier and more accurate if we already have DOIs. Here start by assigning our DOIs to a variable `my_dois`, then pass it to `cr_works()` in the `doi` argument:

```
my_references_dois <- c("10.2139/ssrn.2697412", "10.1016/j.joi.2016.08.002", "10.1371/journal.pone.00209
my_references_dois_works <- rcrossref::cr_works(doi = my_references_dois) %>%
  pluck("data")
my_references_dois_works
```

```
## # A tibble: 7 x 31
##   container.title created deposited doi   indexed issn  issued member
##   <chr>           <chr>   <chr>     <chr> <chr>   <chr> <chr>  <chr>
## 1 SSRN Electroni~ 2015-1~ 2015-12-~ 10.2~ 2019-0~ 1556~ 2014   78
## 2 Journal of Inf~ 2016-0~ 2018-09-~ 10.1~ 2019-0~ 1751~ 2016-~ 78
## 3 PLoS ONE        2011-0~ 2018-10-~ 10.1~ 2019-0~ 1932~ 2011-~ 340
## 4 Frontiers in P~ 2018-0~ 2018-08-~ 10.3~ 2019-0~ 1664~ 2018-~ 1965
## 5 Nature          2018-0~ 2018-01-~ 10.1~ 2019-0~ 0028~ 2018-~ 297
## 6 F1000Research   2016-0~ 2018-12-~ 10.1~ 2019-0~ 2046~ 2016-~ 2560
## 7 <NA>            2019-0~ 2019-01-~ 10.7~ 2019-0~ <NA>  2012   281
## # ... with 23 more variables: prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, author <list>, alternative.id <chr>,
## #   published.print <chr>, issue <chr>, page <chr>, update.policy <chr>,
## #   volume <chr>, assertion <list>, link <list>, license <list>,
## #   published.online <chr>, subject <chr>, reference <list>,
## #   abstract <chr>, isbn <chr>
```

Getting citation data with `cr_citation_count`

Citation counts per article are not returned with `cr_journals`, but you can get them with `cr_citation_count`. We will create first a vector of DOIs `my_references_dois`, then pass them to `cr_citation_count()`:

```
my_references_dois <- c("10.2139/ssrn.2697412", "10.1016/j.joi.2016.08.002", "10.1371/journal.pone.00209
my_references_citation_count <- rcrossref::cr_citation_count(doi = my_references_dois)
my_references_citation_count
```

```
##                              doi count
## 1           10.2139/ssrn.2697412     4
## 2      10.1016/j.joi.2016.08.002    16
## 3   10.1371/journal.pone.0020961   204
## 4       10.3389/fpsyg.2018.01487     0
## 5     10.1038/d41586-018-00104-7     2
## 6  10.12688/f1000research.8460.2     2
## 7 10.7551/mitpress/9286.001.0001    42
```

Then we can join it to the full data frame with `left_join()` from the `dplyr` package. We'll go ahead and sort by the citation count (descending from highest to lowest) using `arrange()` from `dplyr`:

```
my_references_works_citation_count_joined <- my_references_dois_works %>%
  left_join(my_references_citation_count, by = "doi") %>%
  arrange(desc(count))
my_references_works_citation_count_joined
```

```
## # A tibble: 7 x 32
##   container.title created deposited doi   indexed issn  issued member
##   <chr>           <chr>   <chr>     <chr> <chr>   <chr> <chr>  <chr>
## 1 PLoS ONE        2011-0~ 2018-10-~ 10.1~ 2019-0~ 1932~ 2011-~ 340
## 2 <NA>            2019-0~ 2019-01-~ 10.7~ 2019-0~ <NA>  2012   281
```

```
## 3 Journal of Inf~ 2016-0~ 2018-09-~ 10.1~ 2019-0~ 1751~ 2016-~ 78
## 4 SSRN Electroni~ 2015-1~ 2015-12-~ 10.2~ 2019-0~ 1556~ 2014   78
## 5 Nature          2018-0~ 2018-01-~ 10.1~ 2019-0~ 0028~ 2018-~ 297
## 6 F1000Research   2016-0~ 2018-12-~ 10.1~ 2019-0~ 2046~ 2016-~ 2560
## 7 Frontiers in P~ 2018-0~ 2018-08-~ 10.3~ 2019-0~ 1664~ 2018-~ 1965
## # ... with 24 more variables: prefix <chr>, publisher <chr>,
## #   reference.count <chr>, score <chr>, source <chr>, title <chr>,
## #   type <chr>, url <chr>, author <list>, alternative.id <chr>,
## #   published.print <chr>, issue <chr>, page <chr>, update.policy <chr>,
## #   volume <chr>, assertion <list>, link <list>, license <list>,
## #   published.online <chr>, subject <chr>, reference <list>,
## #   abstract <chr>, isbn <chr>, count <dbl>
```

Unfortunately, only the publishers who are members of Crossref and are owners of the target article are able to retrieve the actual articles citing it, as described in this article. Until recently we've been beholden to Web of Science or Scopus for that data–unless you find some sneaky way of scraping it–but the Initiative for Open Citations is making great progress towards "the unrestricted availability of scholarly citation data."

Getting citation data with `cr_cn()`

As we discussed briefly above, we can use `cr_cn()` to input a set of DOIs and get back the citation data in multiple formats. I am not talking about a list of articles that are cited in a given bibliography, nor what articles have cited the article in question, but rather a styled reference that you can input into a bibliography.

We first create our vector of dois:

```
my_references_dois <- c("10.2139/ssrn.2697412", "10.1016/j.joi.2016.08.002", "10.1371/journal.pone.00209
```

Getting formatted references in a text file

We can use the `cr_cn()` function from the `rcrossref` package to get the citations to those articles in text form in the style you specify. We'll put it into Chicago:

```
my_citations <- rcrossref::cr_cn(my_references_dois,
                                 format = "text",
                                 style = "chicago-note-bibliography")
```

This returns each citation into a list element. We can use the `map_chr` and the `pluck` functions from `purrr` to instead assign them to a character vector. We can just overwrite the old one.

```
my_citations <- my_citations %>%
  purrr::map_chr(., purrr::pluck, 1)
my_citations
```

```
## [1] "Frosio, Giancarlo F. "Open Access Publishing: A Literature Review." SSRN Electronic Journal (201
## [2] "Laakso, Mikael, and Bo-Christer Björk. "Hybrid Open access-A Longitudinal Study." Journal of Inf
## [3] "Laakso, Mikael, Patrik Welling, Helena Bukvova, Linus Nyman, Bo-Christer Björk, and Turid Hedlun
## [4] "Paulus, Frieder M., Nicole Cruz, and Sören Krach. "The Impact Factor Fallacy." Frontiers in Psyc
## [5] "Shotton, David. "Funders Should Mandate Open Citations." Nature 553, no. 7687 (January 11, 2018)
## [6] "Tennant, Jonathan P., François Waldner, Damien C. Jacques, Paola Masuzzo, Lauren B. Collister, a
## [7] "Suber, Peter. "Open Access" (2012). doi:10.7551/mitpress/9286.001.0001."
```

Beautiful formatted citations from simply a list of DOIs! You can then write this to a text file. We set our `my_filepath` variable way up towards the top of this document, if you want to change it.

```
write(my_citations, file = file.path(my_filepath, "my_citations.txt"))
```

The above is helpful if you need to paste the references somewhere, and there are loads of other citation styles included in `rcrossref`–view them by calling `rcrossref::get_styles()` and it will print the list to your console. I'll just print the first 15 below:

x
academy-of-management-review
accident-analysis-and-prevention
aci-materials-journal
acm-sig-proceedings-long-author-list
acm-sig-proceedings
acm-sigchi-proceedings-extended-abstract-format
acm-sigchi-proceedings
acm-siggraph
acme-an-international-journal-for-critical-geographies
acta-amazonica
acta-anaesthesiologica-scandinavica
acta-anaesthesiologica-taiwanica
acta-chirurgiae-orthopaedicae-et-traumatologiae-cechoslovaca
acta-naturae
acta-neurobiologiae-experimentalis

Getting formatted references in a BibTeX or RIS file

In addition to a text file, you can also write it to BibTeX or RIS:

```
my_citations_bibtex <- rcrossref::cr_cn(my_references_dois, format = "bibtex") %>%
  purrr::map_chr(., purrr::pluck, 1)
```

Write it to a .bib file using `write_lines()` from the `readr` package. Remember to replace MyUserName with your own user name, which you can get from running `Sys.getenv("USERNAME")`. Or just create a directory and use that.

```
readr::write_lines(my_citations_bibtex, file.path(my_filepath, "my_citations_bibtex.bib"))
```

Same with RIS files. EndNote has a hard time reading BibTeX, so do this if you use that as your reference management software. Instead, set the format to RIS. For this to work, we must first make it into a `tibble`:

```
my_citations_ris <- rcrossref::cr_cn(my_references_dois, format = "ris") %>%
  purrr::map_chr(., purrr::pluck, 1) %>%
  tibble::tibble()
```

Use `write_csv()` from `readr` to write the RIS file.

```
readr::write_csv(my_citations_ris, file.path(my_filepath, "my_citations_ris.ris"))
```

RStudio Add-In

As described in the documentation, `rcrossref` installs an add-in to RStudio:

> On installation of rcrossref you get an RStudio Addin. To use the Addin, go to the top toolbar > Tools > Addins > Add Crossref Citations. You'll get a window pop up that you can put in DOIs for.

If the DOI is found, the bibtex citations will be added to a file called crossref.bib. New citations will be appended to that file. Addin authored by Hao Zhu (https://github.com/haozhu233)

Check out the `RefManageR` package for more you can do with citation files in R.

Using `roadoi` to check for open access

`roadoi` was developed by Najko Jahn, with reviews from Tuija Sonkkila and Ross Mounce. It interfaces with Unpaywall (which used to be called oaDOI), an important tool developed by ImpactStory (Heather Piwowar and Jason Priem) for locating open access versions of scholarship–read more in this *Nature* article. See here for the `roadoi` documentation.

This incredible Introduction to `roadoi` by Najko Jahn provides much of what you need to know to use the tool, as well as an interesting use case. Also see his recently published article Open Access Evidence in Unpaywall, running deep analysis on Unpaywall data.

First install the package and load it.

```
install.packages("roadoi")
library(roadoi)
```

Setting up `roadoi`

As with `rcrossref`, your API calls to Unpaywall must include a valid email address where you can be reached in order to keep the service open and free for everyone.

Run this line of code, replacing the example with your email address:

```
options(roadoi_email = "name@example.com")
```

Your email address will now be shared with Unpaywall.

Checking OA status with `oadoi_fetch`

We then create DOI vector and use the `oadoi_fetch()` function from `roadoid`.

```
my_references_dois <- c("10.2139/ssrn.2697412", "10.1016/j.joi.2016.08.002", "10.1371/journal.pone.0020
my_reference_dois_oa <- roadoi::oadoi_fetch(dois = my_references_dois)
my_reference_dois_oa
```

```
## # A tibble: 7 x 16
##   doi   best_oa_location oa_locations data_standard is_oa genre
##   <chr> <list>           <list>               <int> <lgl> <chr>
## 1 10.2~ <tibble [0 x 0]> <tibble [0 ~             2 FALSE jour~
## 2 10.1~ <tibble [1 x 8]> <tibble [1 ~             2 TRUE  jour~
## 3 10.1~ <tibble [1 x 9]> <tibble [5 ~             2 TRUE  jour~
## 4 10.3~ <tibble [1 x 9]> <tibble [4 ~             2 TRUE  jour~
## 5 10.1~ <tibble [1 x 8]> <tibble [1 ~             2 TRUE  jour~
## 6 10.1~ <tibble [1 x 9]> <tibble [6 ~             2 TRUE  jour~
## 7 10.7~ <tibble [0 x 0]> <tibble [0 ~             2 FALSE book
## # ... with 10 more variables: journal_is_oa <lgl>,
## #   journal_is_in_doaj <lgl>, journal_issns <chr>, journal_name <chr>,
## #   publisher <chr>, title <chr>, year <chr>, updated <chr>,
## #   non_compliant <list>, authors <list>
```

The returned variables are described on the Unpaywall Data Format page.

We can see that Unpaywall could not find OA versions for two of the seven of these, so we will filter them out:

```
my_reference_dois_oa <- my_reference_dois_oa %>%
  filter(is_oa == TRUE)
```

You can also use `roadoi` to run some analysis on these, looking at whether it is green, gold or hybrid OA, the license, etc.

Getting URLs

I will copy and paste code directly from Najko Jahn to extract the URLs:

```
my_reference_dois_oa <- my_reference_dois_oa %>%
  dplyr::mutate(
    urls = purrr::map(best_oa_location, "url") %>%
                  purrr::map_if(purrr::is_empty, ~ NA_character_) %>%
                  purrr::flatten_chr()
                )
my_reference_dois_oa$urls
```

```
## [1] "https://doi.org/10.1016/j.joi.2016.08.002"
## [2] "https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0020961&type=printable"
## [3] "https://www.frontiersin.org/articles/10.3389/fpsyg.2018.01487/pdf"
## [4] "https://www.nature.com/magazine-assets/d41586-018-00104-7/d41586-018-00104-7.pdf"
## [5] "https://f1000research.com/articles/5-632/v2/pdf"
```

Check out the `fulltext` package to actually download the papers directly from within R.

RStudio Add-In

As described in the documentation, on installation of `roadoi` you get an RStudio Addin. To use the Addin, go to the top toolbar > Tools > Addins > roadoi > Execute. The addin works as follows:

1. Copy up to ten line-separated DOIs into the text area
2. Press the button "Run!"
3. Click on the links in the table to download full-text

Conclusion

The Crossref and Unpaywall APIs are excellent tools for analyzing research activity on multiple levels. `rcrossref` and `roadoi` make gathering and cleaning the data easier. Thanks to both ORCID and the rOpenSci team for their contributions to the community.