5 APIs and Packages

R. Heller · Date: 2026-05-09 · Last changed: 2026-10-31

5.1 Learning objectives

After completing this chapter, you will be able to:

- Query the OpenAlex API using `openalexR` to retrieve works, authors, institutions, and concepts
- Apply filters, pagination, and sampling to control the size and scope of API results
- Deduplicate records by DOI and clean author and affiliation metadata
- Cache API responses locally so repeated book builds do not re-query the API
- Use `rcrossref` and `pubmedR` as complementary data sources

5.2 Setup

```
library(tidyverse)
library(openalexR)
library(glue)

set.seed(20260509)

source(here::here("R", "api_helpers.R"))
source(here::here("R", "utils.R"))
source(here::here("R", "sci_palette.R"))
```

5.3 Conceptual background

Bibliometric research begins with data. The quality and completeness of a bibliographic corpus determines the validity of every downstream analysis — from citation counts to network maps to topic models.

Historically, Web of Science and Scopus have dominated as data sources for scientometric research. Both require institutional subscriptions and impose restrictions on bulk data use. The launch of **OpenAlex** in 2022 changed this landscape fundamentally (Priem et al. 2022). OpenAlex is a fully open index of over 250 million scholarly works, built from Crossref, PubMed, institutional repositories, and other open sources. It provides a free REST API with no authentication required, making it the ideal backbone for reproducible bibliometric research.

OpenAlex organises its data around five entity types: **works** (publications), **authors**, **sources** (journals, repositories), **institutions**, and **topics** (hierarchical subject tags). Each entity has a unique OpenAlex ID and can be queried via filters on metadata fields such as publication date, citation count, open-access status, and institutional affiliation.

The `openalexR` package provides a tidy R interface to the OpenAlex API. It handles pagination, rate limiting, and result parsing, returning clean tibbles ready for analysis. For metadata not covered by OpenAlex — such as reference lists and funding information — `rcrossref` provides access to Crossref's extensive metadata registry, and `pubmedR` covers the biomedical literature indexed in PubMed/MEDLINE.

5.4 Worked example

We build a small corpus of recent scientometrics research, demonstrating the full acquisition-to-clean-data pipeline.

5.4.1 Querying works from OpenAlex

The provider function is `openalexR::oa_fetch()`. Runnable examples use the book's `fetch_openalex()` wrapper, sourced above, to cache responses by their exact arguments. Set `SCIENTOMETRICS_OFFLINE=1` to reuse existing OpenAlex caches without requesting new data; a missing cache stops with an explicit error. Other providers have their own network requirements. We search for works published in the journal *Scientometrics* between 2020 and 2023, sampling 200 records for a manageable demonstration.

```
works ← fetch_openalex(
  entity = "works",
  primary_location.source.id = "S148561398",
  from_publication_date = "2020-01-01",
  to_publication_date = "2023-12-31",
  options = list(sample = 200, seed = 42)
)

glimpse(works)
```

```
#> Rows: 200
#> Columns: 45
#> $ id                              <chr> "https://openalex.org/W3036497540", "htt…
#> $ title                           <chr> "The world-systemic network of global el…
#> $ display_name                    <chr> "The world-systemic network of global el…
#> $ authorships                     <list> [<tbl_df[2 x 8]>], [<tbl_df[6 x 8]>], […
#> $ abstract                        <chr> "In this paper, we analyze the education…
#> $ doi                             <chr> "https://doi.org/10.1007/s11192-020-0356…
#> $ publication_date                <date> 2020-06-17, 2022-12-17, 2022-11-30, 202…
#> $ publication_year                <int> 2020, 2022, 2022, 2023, 2021, 2021, 2021…
#> $ relevance_score                 <dbl> 1.000, 1.000, 0.999, 0.999, 0.998, 0.998…
#> $ fwci                            <dbl> 5.114, 2.485, 0.274, 49.658, 1.440, 3.19…
#> $ cited_by_count                  <int> 40, 64, 1, 199, 10, 19, 9, 31, 15, 50, 1…
#> $ counts_by_year                  <list> [<data.frame[7 x 2]>], [<data.frame[5 x…
#> $ ids                             <list> <"https://openalex.org/W3036497540", "h…
#> $ type                            <chr> "article", "article", "article", "articl…
#> $ is_oa                           <lgl> TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, …
#> $ is_oa_anywhere                  <lgl> TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, TR…
#> $ oa_status                       <chr> "hybrid", "green", "green", "hybrid", "c…
#> $ oa_url                          <chr> "https://link.springer.com/content/pdf/1…
#> $ any_repository_has_fulltext     <lgl> TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, TR…
#> $ source_display_name             <chr> "Scientometrics", "Scientometrics", "Sci…
#> $ source_id                       <chr> "https://openalex.org/S148561398", "http…
#> $ issn_l                          <chr> "0138-9130", "0138-9130", "0138-9130", "…
#> $ host_organization               <chr> "https://openalex.org/P4310320108", "htt…
#> $ host_organization_name          <chr> "Springer Nature (Netherlands)", "Spring…
#> $ landing_page_url                <chr> "https://doi.org/10.1007/s11192-020-0356…
#> $ pdf_url                         <chr> "https://link.springer.com/content/pdf/1…
#> $ license                         <chr> "cc-by", NA, NA, "other-oa", NA, NA, NA,…
#> $ version                         <chr> "publishedVersion", "publishedVersion", …
#> $ referenced_works                <list> <"https://openalex.org/W308615189", "ht…
#> $ referenced_works_count          <int> 84, 45, 4, 54, 44, 58, 7, 24, 64, 97, 56…
#> $ related_works                   <list> <"https://openalex.org/W3107920103", "h…
#> $ concepts                        <list> [<data.frame[18 x 5]>], [<data.frame[20…
#> $ topics                          <list> [<tbl_df[12 x 5]>], [<tbl_df[12 x 5]>],…
#> $ keywords                        <list> [<data.frame[8 x 3]>], [<data.frame[6 x…
#> $ is_paratext                     <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE…
#> $ is_retracted                    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE…
#> $ language                        <chr> "en", "en", "en", "en", "en", "en", "en"…
#> $ sustainable_development_goals   <list> [<data.frame[2 x 3]>], NA, NA, NA, NA, …
#> $ awards                          <list> NA, <"https://openalex.org/G63352921", …
#> $ funders                         <list> [<data.frame[1 x 3]>], [<data.frame[4 x…
#> $ apc                             <list> [<data.frame[2 x 5]>], [<data.frame[2 x…
#> $ first_page                      <chr> "2469", "1241", "2029", "4007", "3209", …
#> $ last_page                       <chr> "2495", "1263", "2030", "4028", "3226", …
#> $ volume                          <chr> "124", "128", "128", "128", "126", "126"…
#> $ issue                           <chr> "3", "2", "3", "7", "4", "7", "10", "1",…
```

The result is a tibble where each row is a work and columns include `id`, `display_name`, `publication_date`, `cited_by_count`, `doi`, and nested list-columns for authors and concepts.

5.4.2 Inspecting the results

```
works |>
  select(display_name, publication_date, cited_by_count, doi) |>
  arrange(desc(cited_by_count)) |>
  head(10)
```

```
#> # A tibble: 10 × 4
#>    display_name                              publication_date cited_by_count doi
#>    <chr>                                     <date>                     <int> <chr>
#>  1 Google Scholar, Microsoft Academic, Sc… 2020-09-21                  1658 http…
#>  2 Sample size in bibliometric analysis    2020-07-31                   281 http…
#>  3 Rethinking SME default prediction: a s… 2021-01-29                   212 http…
#>  4 Visibility, impact, and applications o… 2023-06-01                   199 http…
#>  5 Preliminary analysis of COVID-19 acade… 2020-06-25                   168 http…
#>  6 International collaboration during the… 2021-02-14                   129 http…
#>  7 Why do papers from international colla… 2022-08-22                   126 http…
#>  8 Coronavirus mapping in scientific publ… 2020-07-01                   124 http…
#>  9 Evolution of research topics in LIS be… 2020-10-10                   112 http…
#> 10 A deep learning based method for extra… 2020-07-24                   105 http…
```

5.4.3 Querying authors

We can also retrieve author-level data. Let us look up authors associated with scientometrics research by filtering on the relevant topic.

```
authors ← fetch_openalex(
  entity = "authors",
  topics.id = "T10102",
  works_count = ">50",
  options = list(sample = 50, seed = 42)
)

authors ▷
  select(display_name, works_count, cited_by_count) ▷
  arrange(desc(cited_by_count)) ▷
  head(10)
```

```
#> # A tibble: 10 × 3
#>    display_name             works_count cited_by_count
#>    <chr>                          <int>          <int>
#>  1 Jennifer Tetzlaff                 80         603386
#>  2 Steve Lawrence                   144          26565
#>  3 Alejandro Valencia-Arías         408           7081
#>  4 Ian Rowlands                     168           5854
#>  5 Alfredo Alcayde                  174           5291
#>  6 Wei Hua Zhou                     284           4151
#>  7 Mathieu Albert                    63           3563
#>  8 Iain Hrynaszkiewicz              144           2061
#>  9 Khaled Moustafa                  156           2048
#> 10 Sai Leung Ng                      70           1923
```

5.4.4 Deduplication by DOI

Real-world corpora often contain duplicates when merging results from multiple queries or sources. The `dedupe_by_doi()` function from our companion package handles this cleanly.

```
works_with_dupes ← bind_rows(works, works[1:10, ])
cat(glue("Before dedup: {nrow(works_with_dupes)} rows\n\n"))
```

```
#> Before dedup: 210 rows
```

```
works_clean ← dedupe_by_doi(works_with_dupes)
cat(glue("After dedup:  {nrow(works_clean)} rows\n"))
```

```
#> After dedup:  200 rows
```

5.4.5 Caching for reproducible builds

The `fetch_openalex()` wrapper in `R/api_helpers.R` automatically caches API responses to disk. On subsequent calls with the same arguments, it reads from cache instead of hitting the API.

```
works_cached ← fetch_openalex(
  entity = "works",
  primary_location.source.id = "S148561398",
  from_publication_date = "2023-01-01",
  to_publication_date = "2023-06-30",
  options = list(sample = 50, seed = 42),
  cache_dir = "_freeze/openalex_cache",
  cache_days = 30
)

nrow(works_cached)
```

```
#> [1] 50
```

5.4.6 Complementary sources: Crossref

For detailed metadata (reference lists, funders, licenses), Crossref is invaluable.

```
# Requires network access — cached output used in book builds
cr_work ← rcrossref::cr_works(doi = "10.1007/s11192-017-2300-7")
cr_work$data ▷
  select(title, container.title, deposited, is.referenced.by.count)
```

5.4.7 Visualization

```
works ▷
  mutate(year = year(publication_date)) ▷
  count(year) ▷
  ggplot(aes(x = year, y = n)) +
  geom_col(fill = palette_sci(1)) +
  labs(x = "Publication year", y = "Number of works") +
  theme_sci()
```

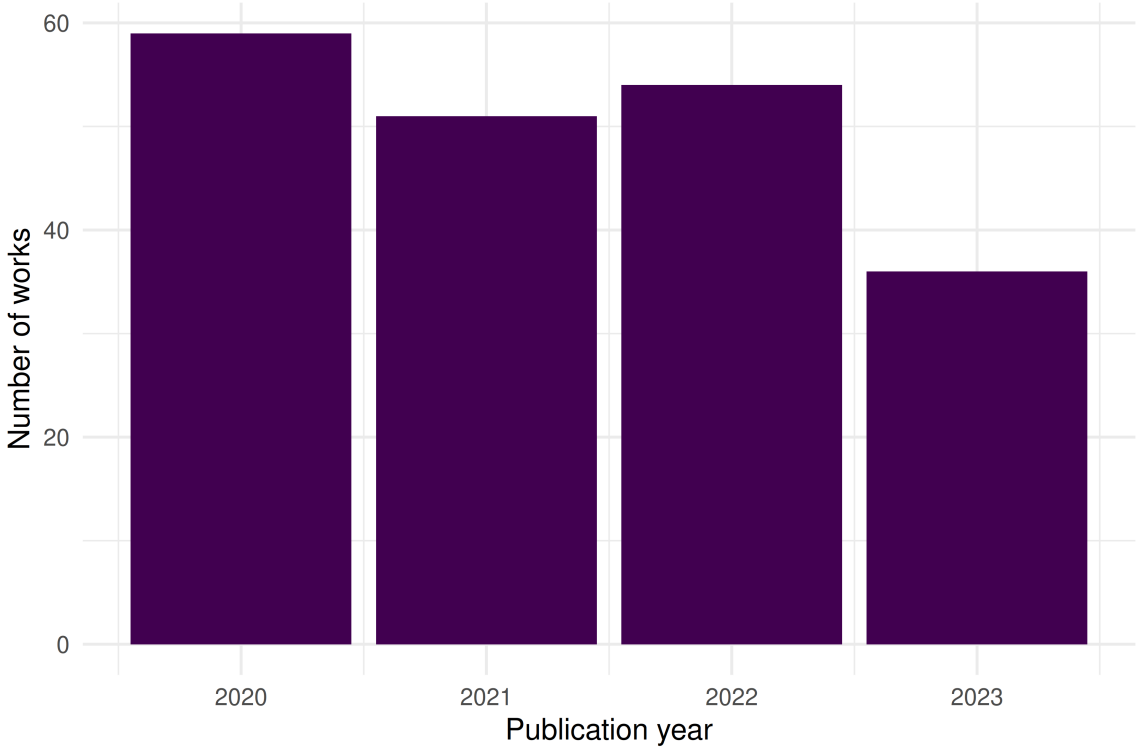


Figure 5.1: Annual publication counts for sampled Scientometrics articles (2020–2023).

5.5 Diagnostics and interpretation

When working with API data, always verify:

- **Completeness**: Compare your result count against the API's reported `meta$count` to confirm you received all expected records. With sampling, the count reflects the sample, not the full set.
- **Coverage dates**: Check `min(works$publication_date)` and `max(works$publication_date)` to confirm your date filters were applied correctly.
- **DOI presence**: Not all works have DOIs. Check `mean(!is.na(works$doi))` to understand the DOI coverage rate in your corpus.
- **Nested columns**: Author and concept columns are list-columns. Use `tidyr::unnest()` carefully and inspect for `NULL` entries before unnesting.

```
cat(glue("Total works: {nrow(works)}\n"))
```

```
#> Total works: 200
```

```
cat(glue("DOI coverage: {scales::percent(mean(!is.na(works$doi)))}\n"))
```

```
#> DOI coverage: 100%
```

```
cat(glue("Date range: {min(works$publication_date)} to
{max(works$publication_date)}\n"))
```

```
#> Date range: 2020-01-11 to 2023-12-18
```

5.6 Limitations and responsible use

5.7 Limitations and responsible use

- **Coverage bias**: OpenAlex is strongest in English-language, peer-reviewed journal articles. Conference proceedings, books, and non-English literature may be underrepresented. Always report the data source and its known coverage limitations.
- **Metadata quality**: Author names and affiliations are parsed algorithmically and may contain errors, especially for names with diacritics or non-Latin scripts.
- **Temporal lag**: OpenAlex updates continuously but may lag behind primary databases by days or weeks. Very recent publications may be missing.
- **API rate limits**: OpenAlex allows 10 requests per second for unauthenticated users (100/s with a polite pool email). Respect these limits to avoid being throttled.
- **Reproducibility**: API results can change as OpenAlex updates its data. Always cache results and document the date of data collection for reproducibility (Hicks et al. 2015).

5.8 Common pitfalls

5.9 Common pitfalls

- **Not caching**: Running `oa_fetch()` directly in code chunks without caching means every `quarto render` hits the API, producing different results each time and risking rate limits.
- **Ignoring pagination**: `oa_fetch()` handles pagination automatically, but very large queries (>10,000 works) can time out. Use filters to narrow your query or use cursor-based pagination.

- **Confusing sampling with filtering**: `options = list(sample = 200)` returns a random sample from the full result set, not the first 200 records. This is useful for demonstrations but not for exhaustive analyses.
- **Treating OpenAlex IDs as permanent**: While generally stable, OpenAlex IDs can change when entities are merged. Use DOIs as the primary identifier for works.
- **Merging sources without deduplication**: When combining results from OpenAlex and Crossref, records may overlap. Always deduplicate by DOI before analysis.

5.10 Exercises

1. **Fetch works by institution.** Use `oa_fetch()` to retrieve works affiliated with your own institution. How many works does OpenAlex index for it? What is the DOI coverage rate? *(Hint: find your institution's OpenAlex ID via `oa_fetch(entity = "institutions", search = "...")`)*

2. **Compare Crossref metadata.** Pick a DOI from your fetched works and retrieve its metadata from Crossref using `rcrossref::cr_works()`. Compare the citation count reported by OpenAlex vs. Crossref. Why might they differ?

3. **Build a multi-year corpus.** Fetch all works from a journal of your choice for 2015–2023 (without sampling). Plot the annual publication count and the median citation count per year. What trends do you observe?

5.11 Solutions

Solutions are provided in E.

5.12 Further reading

- Priem et al. (2022) — The OpenAlex paper; describes the data model, coverage, and API design.
- Aria and Cuccurullo (2017) — The `bibliometrix` package; an alternative high-level interface for bibliometric data acquisition and analysis.
- Garfield (1955) — The foundational paper on citation indexing, motivating why these databases exist.

5.13 Session info

```
sessionInfo()
```

```
#> R version 4.6.1 (2026-06-24)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Ubuntu 24.04.5 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.26.so;  LAPACK
version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
#>  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
#>  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
#> [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: UTC
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] rcrossref_1.2.1 gt_1.4.0        tidytext_0.4.3  glue_1.8.1
#>  [5] openalexR_3.1.2 lubridate_1.9.5 forcats_1.0.1   stringr_1.6.0
#>  [9] dplyr_1.2.1     purrr_1.2.2     readr_2.2.0     tidyr_1.3.2
#> [13] tibble_3.3.1    ggplot2_4.0.4   tidyverse_2.0.0
#>
#> loaded via a namespace (and not attached):
#>  [1] tidyselect_1.2.1   viridisLite_0.4.3  farver_2.1.2       urltools_1.7.3.1
#>  [5] viridis_0.6.5      S7_1.0.0           fastmap_1.2.0      janeaustenr_1.0.0
#>  [9] promises_1.5.0     digest_0.6.39      timechange_0.4.0   mime_0.13
#> [13] lifecycle_1.0.5    tokenizers_0.3.0   magrittr_2.0.5     compiler_4.6.1
#> [17] rlang_1.3.0        sass_0.5.0         tools_4.6.1        utf8_1.2.6
#> [21] yaml_2.3.12        knitr_1.52         labeling_0.4.3     stopwords_2.3
#> [25] htmlwidgets_1.6.4  curl_8.1.0         here_1.0.2         plyr_1.8.9
#> [29] xml2_1.6.0         RColorBrewer_1.1-3 httpcode_0.3.0     miniUI_0.1.2
#> [33] withr_3.0.3        triebeard_0.4.1    grid_4.6.1         xtable_1.8-8
#> [37] scales_1.4.0       crul_1.6.0         cli_3.6.6          rmarkdown_2.32
#> [41] generics_0.1.4     otel_0.3.0         rstudioapi_0.19.0  httr_1.4.9
#> [45] tzdb_0.5.0         cachem_1.1.0       vctrs_0.7.3        Matrix_1.7-5
#> [49] jsonlite_2.0.0     bookdown_0.48      hms_1.1.4          jquerylib_0.1.4
#> [53] DT_0.34.0          stringi_1.8.9      gtable_0.3.6       later_1.4.8
#> [57] downlit_0.4.5      pillar_1.11.1      htmltools_0.5.9    R6_2.6.1
#> [61] rprojroot_2.1.1    evaluate_1.0.5     shiny_1.15.0       lattice_0.22-9
#> [65] SnowballC_0.7.1    memoise_2.0.1      httpuv_1.6.17      bslib_0.12.0
#> [69] Rcpp_1.1.2         gridExtra_2.3.1    xfun_0.62          fs_2.1.0
#> [73] pkgconfig_2.0.3
```