fulltext manual

2017-12-19 - fulltext v0.1.9.9621

Contents

Chapter 1

fulltext manual

> An R package to search across and get full text for open access journals

The `fulltext` package makes it easy to do text-mining by supporting the following steps:

- Search for articles
- Fetch articles
- Get links for full text articles (xml, pdf)
- Extract text from articles / convert formats
- Collect bits of articles that you actually need
- Download supplementary materials from papers

1.1 Info

- Code: https://github.com/ropensci/fulltext/
- Issues: https://github.com/ropensci/fulltext/issues
- CRAN: https://cran.rstudio.com/web/packages/fulltext/

1.2 Citing fulltext

> Scott Chamberlain & Will Pearse (2017). fulltext: Full Text of 'Scholarly' Articles Across Many Data Sources. R package version 0.1.9.9621. https://github.com/ropensci/fulltext

1.3 Installation

Stable version from CRAN

```r
install.packages("fulltext")
```

Development version from GitHub

```r
devtools::install_github("ropensci/fulltext")
```

Load library

```r
library('fulltext')
```

Chapter 2

Introduction

2.1 User interface

Functions in `fulltext` are setup to make the package as easy to use as possible. The functions are organized around use cases:

- Search for articles
- Get full text links
- Get articles
- Get abstracts
- Pull out article sections of interest

Because there are so many data sources for scholarly texts, it makes a lot of sense to simplify the details of each data source, and present a single user interface to all of them.

Chapter 3

Data sources

Data sources in `fulltext` include:

- Crossref - via the `rcrossref` package
- Public Library of Science (PLOS) - via the `rplos` package
- Biomed Central
- arXiv - via the `aRxiv` package
- bioRxiv - via the `biorxivr` package
- PMC/Pubmed via Entrez - via the `rentrez` package
- Many more are supported via the above sources (e.g., *Royal Society Open Science* is available via Pubmed)
- We **will** add more, as publishers open up, and as we have time…See the master list here

Chapter 4

Authentication

Some data sources require authentication. Here's a breakdown of how to do authentication by data source:

- **BMC**: BMC is integrated into Springer Publishers now, and that API requires an API key. Get your key by signing up at https://dev.springer.com/, then you'll get a key. Pass the key to a named parameter `key` to `bmcopts`. Or, save your key in your `.Renviron` file as `SPRINGER_KEY`, and we'll read it in for you, and you don't have to pass in anything.
- **Scopus**: Scopus requires an API key to search their service. Go to https://dev.elsevier.com/index.html, register for an account, then when you're in your account, create an API key. Pass in as variable `key` to `scopusopts`, or store your key under the name `ELSEVIER_SCOPUS_KEY` as an environment variable in `.Renviron`, and we'll read it in for you. See `?Startup` in R for help.
- **Microsoft**: Get a key by creating an Azure account at https://www.microsoft.com/cognitive-services/en-us/subscriptions, then requesting a key for **Academic Knowledge API** within **Cognitive Services**. Store it as an environment variable in your `.Renviron` file - see [Startup] for help. Pass your API key into `maopts` as a named element in a list like `list(key = Sys.getenv('MICROSOFT_ACADEMIC_KEY'))`
- **Crossref**: Crossref encourages requests with contact information (an email address) and will forward you to a dedicated API cluster for improved performance when you share your email address with them. https://github.com/CrossRef/rest-api-doc#good-manners--more-reliable-service To pass your email address to Crossref via this client, store it as an environment variable in `.Renviron` like `crossref_email = name@example.com`

None needed for **PLOS**, **eLife**, **arxiv**, **biorxiv**, **Euro PMC**, or **Entrez** (though soon you will get better rate limtits with auth for Entrez)

Chapter 5

Search

Search is what you'll likely start with for a number of reasons. First, search functionality in `fulltext` means that you can start from searching on words like 'ecology' or 'cellular' - and the output of that search can be fed downstream to the next major task: fetching articles.

5.1 Usage

```r
library(fulltext)
```

List backends available

```r
ft_search_ls()
```

```
#> [1] "arxiv"      "biorxivr"   "bmc"        "crossref"   "entrez"
#> [6] "europe_pmc" "ma"         "plos"       "scopus"
```

Search - by default searches against PLOS (Public Library of Science)

```r
res <- ft_search(query = "ecology")
```

The output of `ft_search` is a `ft` S3 object, with a summary of the results:

```r
res
```

```
#> Query:
#>   [ecology]
#> Found:
#>   [PLoS: 41094; BMC: 0; Crossref: 0; Entrez: 0; arxiv: 0; biorxiv: 0; Europe PMC: 0; Scopus: 0; Micr
#> Returned:
#>   [PLoS: 10; BMC: 0; Crossref: 0; Entrez: 0; arxiv: 0; biorxiv: 0; Europe PMC: 0; Scopus: 0; Microso
```

and has slots for each data source:

```r
names(res)
```

```
#> [1] "plos"     "bmc"      "crossref" "entrez"   "arxiv"    "biorxiv"
#> [7] "europmc"  "scopus"   "ma"
```

Get data for a single source

```r
res$plos
```

```
#> Query: [ecology]
#> Records found, returned: [41094, 10]
#> License: [CC-BY]
#>                              id
#> 1  10.1371/journal.pone.0001248
#> 2  10.1371/journal.pone.0059813
#> 3  10.1371/journal.pone.0155019
#> 4  10.1371/journal.pone.0080763
#> 5  10.1371/journal.pone.0150648
#> 6  10.1371/journal.pcbi.1003594
#> 7  10.1371/journal.pone.0102437
#> 8  10.1371/journal.pone.0175014
#> 9  10.1371/journal.pone.0166559
#> 10 10.1371/journal.pone.0054689
```

Chapter 6

Links

links

Chapter 7

Fetch

fetch

Chapter 8

Chunks

chunks

Chapter 9

Supplementary

supplementary

Chapter 10

Use cases

use cases

Chapter 11

Literature

Here is a review of existing methods.

Bibliography