Package 'fulltext'

August 6, 2015

Title Full Text of 'Scholarly' Articles Across Many Data Sources

Description Provides a single interface to many sources of full text 'scholarly' data, including 'Biomed Central', Public Library of Science, 'Pubmed Central', 'eLife', 'F1000Research', 'PeerJ', 'Pensoft', 'Hindawi', 'arXiv' 'preprints', and more. Functionality included for searching for articles, downloading full or partial text, converting to various data formats used in and outside of R.

Version 0.1.0

License MIT + file LICENSE

URL https://github.com/ropensci/fulltext

BugReports https://github.com/ropensci/fulltext/issues

LazyLoad yes

VignetteBuilder knitr

Imports methods, utils, stats, httr (>= 1.0.0), magrittr, xml2, jsonlite, rplos (>= 0.5.2), rcrossref, aRxiv, rentrez, tm, rredis, R.cache, digest, whisker

Suggests roxygen2, testthat, knitr

NeedsCompilation no

Author Scott Chamberlain [aut, cre]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2015-08-06 21:58:33

R topics documented:

fulltext-package 2
cache 3
chunks 4
collect 6
ft_browse 7

ft_extract 8
ft_extract_corpus 9
ft_get 10
ft_providers 14
ft_search 15
ft_serialize 16
pdfx 18

Index **19**

`fulltext-package` *Fulltext search and retrieval of scholarly texts.*

Description

fulltext is a single interface to many sources of scholarly texts. In practice, this means only ones that are legally useable. We will support sources that require authentication on a case by case basis - that is, if more than just a few people will use it, and it's not too burdensome to include, then we can include that source.

What's included

We currently include support for search and full text retrieval for a variety of publishers. See `ft_search` for what we include for search, and `ft_get` for what we include for full text retrieval.

Use cases

The following are tasks/use cases supported:

- search - `ft_search`
- get texts - `ft_get`
- extract text from pdfs - `ft_extract`
- serialize to different data formats - `ft_serialize`
- extract certain article sections (e.g., authors) - `chunks`

DOI delays

Beware that DOIs are not searchable via Crossref/Entrez immediately. The delay may be as much as a few days, though should be less than a day. This delay should become shorter as services improve. The point of this is that you man not find a match for a relatively new DOI (e.g., for an article published the same day). We've tried to account for this for some publishers. For example, for Crossref we search Crossref for a match for a DOI, and if none is found we attempt to retrieve the full text from the publisher directly.

Feedback

Let us know what you think at `https://github.com/ropensci/fulltext/issues`

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

---

cache *Cache blobs of json, xml or pdfs of text from ft_get() function*

Description

Cache blobs of json, xml or pdfs of text from ft_get() function

Usage

```
cache_options_set(cache = TRUE, backend = "rds", path = "~/.fulltext")

cache_options_get()

cache_clear(cachetype = NULL)
```

Arguments

cache	(logical) If TRUE, cache results, if not objects saved within R session.
backend	(character) One of rds, rcache, redis
path	path to local storage. used only if backend="rds"
cachetype	The cache type

Examples

```
## Not run:
ft_get('10.1371/journal.pone.0086169', from='plos', cache=FALSE)
ft_get('10.1371/journal.pone.0086169', from='plos', cache=TRUE)

cache_options_set(backend="redis")
cache_options_get()
(x <- ft_get('10.1371/journal.pone.0086169', from='plos', cache=TRUE, backend="redis"))
x %>% collect()

cache_options_set(backend="rcache")
cache_options_get()
(x <- ft_get('10.1371/journal.pone.0086169', from='plos'))
x %>% collect()

# Many different sources
(res <- ft_search(query='ecology', from='entrez'))
cache_options_set(backend="rds")
out <- ft_get(res)
out$entrez
out %>% collect() %>% chunks("title")

## End(Not run)
```

chunks *Extract chunks of data from articles*

Description

chunks makes it easy to extract sections of an article. You can extract just authors across all articles, or all references sections, or the complete text of each article. Then you can pass the output downstream for vizualization and analysis.

Usage

```
chunks(x, what = "all")

tabularize(x)
```

Arguments

x	An object of class ft_data, the output from a call to ft_get
what	What to get, can be one or more in a vector or list. See Details.

Details

Options for the what parameter:

- front - Publisher, journal and article metadata elements
- body - Body of the article
- back - Back of the article, acknowledgments, author contributions, references
- title - Article title
- doi - Article DOI
- categories - Publisher's categories, if any
- authors - Authors
- keywords - Keywords
- abstract - Article abstract
- executive_summary - Article executive summary
- refs - References
- refs_dois - References DOIs - if available
- publisher - Publisher name
- journal_meta - Journal metadata
- article_meta - Article metadata
- acknowledgments - Acknowledgments
- permissions - Article permissions
- history - Dates, recieved, published, accepted, etc.

Note that we currently only support PLOS, eLife, and Entrez right now, more to come.

Value

A list of output, one for each thing requested

Examples

```
## Not run:
x <- ft_get('10.1371/journal.pone.0086169', from='plos')
chunks(x, what="authors")

library("rplos")
(dois <- searchplos(q="*:*", fl='id',
   fq=list('doc_type:full',"article_type:\"research article\""), limit=5)$data$id)
x <- ft_get(dois, from="plos")
x %>% chunks("front")
x %>% chunks("body")
x %>% chunks("back")
x %>% chunks("history")
x %>% chunks(c("doi","history")) %>% tabularize()
x %>% chunks("authors")
x %>% chunks(c("doi","categories"))
x %>% chunks("all")
x %>% chunks("publisher")
x %>% chunks("acknowledgments")
x %>% chunks("permissions")
x %>% chunks("journal_meta")
x %>% chunks("article_meta")

# Coerce list output to a data.frame, where possible
(dois <- searchplos(q="*:*", fl='id',
   fq=list('doc_type:full',"article_type:\"research article\""), limit=5)$data$id)
x <- ft_get(dois, from="plos")
x %>% chunks("publisher") %>% tabularize()
x %>% chunks("refs") %>% tabularize()
x %>% chunks(c("doi","publisher")) %>% tabularize()
x %>% chunks(c("doi","publisher","permissions")) %>% tabularize()

x <- ft_get(c("10.3389/fnagi.2014.00130",'10.1155/2014/249309','10.1155/2014/162024'),
   from='entrez')
x %>% chunks("doi") %>% tabularize()
x %>% chunks("authors") %>% tabularize()
x %>% chunks(c("doi","publisher","permissions")) %>% tabularize()
x %>% chunks("history") %>% tabularize()

x <- ft_get('10.3389/fnagi.2014.00130', from='entrez')
x %>% chunks("keywords")

# Piping workflow
opts <- list(fq=list('doc_type:full',"article_type:\"research article\""))
ft_search(query='ecology', from='plos', plosopts = opts)$plos$data$id %>%
 ft_get(from = "plos") %>%
 chunks("publisher")
```

```
# Via entrez
res <- ft_get(c("10.3389/fnagi.2014.00130",'10.1155/2014/249309','10.1155/2014/162024'),
   from='entrez')
chunks(res, what="abstract")
chunks(res, what="title")
chunks(res, what="keywords")
chunks(res, what="publisher")

(res <- ft_search(query='ecology', from='entrez'))
ft_get(res$entrez$data$doi, from='entrez') %>% chunks("title")
ft_get(res$entrez$data$doi[1:4], from='entrez') %>% chunks("acknowledgments")
ft_get(res$entrez$data$doi[1:4], from='entrez') %>% chunks(c('title','keywords'))

# From eLife
x <- ft_get(c('10.7554/eLife.04251', '10.7554/eLife.04986'), from='elife')
x %>% chunks("abstract")
x %>% chunks("publisher")
x %>% chunks("journal_meta")
x %>% chunks("acknowledgments")
x %>% chunks("refs_dois")
x %>% chunks(c("abstract", "executive_summary"))

## End(Not run)
```

`collect` *Collect data from a remote source in fulltext*

Description

`collect` grabs full text data from a remote storage device. `get_text` is a convenience function to grab the nested text data and bring it up in the list for easier access

Usage

```
collect(x, ...)

## S3 method for class 'ft_data'
collect(x, ...)

get_text(x, ...)

## S3 method for class 'ft_data'
get_text(x, ...)
```

Arguments

`x`	Input. An object of class `ft_data`
`...`	Further args, ignored.

Examples

```
## Not run:
# Get some data, stash in rds file
x <- ft_get('10.1371/journal.pone.0086169', from='plos', cache=TRUE, backend="rds")

# note that the data is not in the object, gives NULL
x$plos$data$data

# Collect data from the rds file
y <- x %>% collect()

# note how the data is now in the object
y$plos$data$data

# Let's get the actual
x %>% collect() %>% get_text()

## End(Not run)
```

`ft_browse` *Browse an article in your default browser*

Description

Browse an article in your default browser

Usage

```
ft_browse(x, what = "macrodocs", browse = TRUE)

ft_browse_sections(x, what = "abstract", output = NULL, browse = TRUE)
```

Arguments

`x`	An object of class `ft_data` - the output from a call to `ft_get`
`what`	(character) One of macrodocs (default), publisher, or whisker.
`browse`	(logical) Whether to browse (default) or not. If `FALSE`, return the url.
`output`	A file path, if not given, uses a temporary file, deleted up on leaving the R session.

Details

`what=whisker` not operational yet. When operational, will use whisker to open html page from XML content, each section parsed into separate section.

Examples

```
## Not run:
x <- ft_get('10.7554/eLife.04300', from='elife')
ft_browse(x)
ft_browse(x, browse=FALSE)

ft_browse( ft_get('10.3389/fphar.2014.00109', from="entrez") )

# open to publisher site
ft_browse(x, "publisher")

# Browse sections
x <- ft_get(c('10.1371/journal.pone.0086169',"10.1371/journal.pone.0110535"), from='plos')
ft_browse_sections(x, "abstract")
ft_browse_sections(x, "categories")

opts <- list(fq=list('doc_type:full',"article_type:\"research article\""))
out <- ft_search(query='ecology', from='plos', plosopts = opts)$plos$data$id %>%
 ft_get(from = "plos")
out %>% ft_browse_sections("abstract")
out %>% ft_browse_sections("body")

## End(Not run)
```

`ft_extract` *Extract text from a single pdf document*

Description

`ft_extract` attemps to make it easy to extract text from PDFs, using a variety of extraction tools. Inputs can be either paths to PDF files, or the output of `ft_get` (class `ft_data`).

Usage

```
ft_extract(x, which = "xpdf", ...)

## S3 method for class 'gs_char'
print(x, ...)

## S3 method for class 'xpdf_char'
print(x, ...)
```

Arguments

`x`	Path to a pdf file, or an object of class `ft_data`, the output from `ft_get`
`which`	One of gs or xpdf (default).
`...`	further args passed on

Details

For xpdf, you can pass on addition options via flags. See Examples. Right now, you can't pass options to Ghostscript if you're using the gs option.

Value

An object of class gs_char, xpdf_char

Examples

```
## Not run:
path <- system.file("examples", "example1.pdf", package = "fulltext")

(res_xpdf <- ft_extract(path)) # xpdf is the default
(res_xpdf <- ft_extract(path, "xpdf"))
(res_gs <- ft_extract(path, "gs"))

# pass on options to xpdf
## preserve layout from pdf
ft_extract(path, "xpdf", "-layout")
## preserve table structure as much as possible
ft_extract(path, "xpdf", "-table")
## last page to convert is page 2
ft_extract(path, "xpdf", "-l 2")
## first page to convert is page 3
ft_extract(path, "xpdf", "-f 3")

# use on output of ft_get() to extract pdf to text
## arxiv
res <- ft_get('cond-mat/9309029', from = "arxiv")
res2 <- ft_extract(res)
res$arxiv$data
res2$arxiv$data
res2$arxiv$data$data[[1]]$data

## biorxiv
res <- ft_get('10.1101/012476')
res2 <- ft_extract(res)
res$biorxiv$data
res2$biorxiv$data
res2$biorxiv$data$data[[1]]$data

## End(Not run)
```

`ft_extract_corpus` *Extract text from one to many pdf documents into a tm Corpus or Vcorpus.*

Description

Extract text from one to many pdf documents into a tm Corpus or Vcorpus.

Usage

```
ft_extract_corpus(paths, which = "xpdf", ...)
```

Arguments

paths	Path to one or more pdfs
which	One of gs or xpdf.
...	further args passed on to readerControl parameter in Corpus

Value

A tm Corpus (or VCorpus, later that is)

See Also

ft_extract

Examples

```
## Not run:
path <- system.file("examples", "example1.pdf", package = "fulltext")
(res <- ft_extract_corpus(path, "xpdf"))
tm::TermDocumentMatrix(res$data)

(res_gs <- ft_extract_corpus(path, "gs"))

## End(Not run)
```

---

ft_get *Get full text*

Description

ft_get is a one stop shop to fetch full text of articles, either XML or PDFs. We have specific support for PLOS via the rplos package, Entrez via the rentrez package, and arXiv via the aRxiv package. For other publishers, we have helpers to ft_get to sort out links for full text based on user input. See Details for help on how to use this function.

Usage

```
ft_get(x, from = NULL, plosopts = list(), bmcopts = list(),
  entrezopts = list(), elifeopts = list(), cache = FALSE,
  backend = "rds", path = "~/.fulltext", ...)

## S3 method for class 'character'
ft_get(x, from = NULL, plosopts = list(),
  bmcopts = list(), entrezopts = list(), elifeopts = list(),
  cache = FALSE, backend = "rds", path = "~/.fulltext", ...)

## S3 method for class 'list'
ft_get(x, from = NULL, plosopts = list(), bmcopts = list(),
  entrezopts = list(), elifeopts = list(), cache = FALSE,
  backend = "rds", path = "~/.fulltext", ...)

## S3 method for class 'ft'
ft_get(x, from = NULL, plosopts = list(), bmcopts = list(),
  entrezopts = list(), elifeopts = list(), cache = FALSE,
  backend = "rds", path = "~/.fulltext", ...)
```

Arguments

x	Either identifiers for papers, either DOIs (or other ids) as a list of charcter strings, or a character vector, OR an object of class ft, as returned from ft_search
from	Source to query. Optional.
plosopts	PLOS options. See plos_fulltext
bmcopts	BMC options. See bmc_xml
entrezopts	Entrez options. See entrez_search and entrez_fetch
elifeopts	eLife options
cache	(logical) To cache results or not. If cache=TRUE, raw XML, or other format that article is in is written to disk, then pulled from disk when further manipulations are done on the data. See also cache
backend	(character) One of rds, rcache, or redis
path	(character) Path to local folder
...	Further args passed on to GET

Details

There are various ways to use ft_get:

- Pass in only DOIs - leave from parameter NULL. This route will first query Crossref API for the publisher of the DOI, then we'll use the appropriate method to fetch full text from the publisher. If a publisher is not found for the DOI, then we'll throw back a message telling you a publisher was not found.

- Pass in DOIs (or other pub IDs) and use the `from` parameter. This route means we don't have to make an extra API call to Crossref (thus, this route is faster) to determine the publisher for each DOI. We go straight to getting full text based on the publisher.
- Use `ft_search` to search for articles. Then pass that output to this function, which will use info in that object. This behaves the same as the previous option in that each DOI has publisher info so we know how to get full text for each DOI.

Note that some publishers are available via Entrez, but often not recent articles, where "recent" may be a few months to a year or so. In that case, make sure to specify the publisher, or else you'll get back no data.

Value

An object of class `ft_data` (of type `S3`) with slots for each of the publishers. The returned object is split up by publishers because the full text format is the same within publisher - which should facilitate text mining downstream as different steps may be needed for each publisher's content.

Notes on specific publishers

- arXiv - The IDs passed are not actually DOIs, though they look similar. Thus, there's no way to not pass in the `from` parameter as we can't determine unambiguously that the IDs passed in are from arXiv.org.

Examples

```
## Not run:
# If you just have DOIs and don't know the publisher
## PLOS
ft_get('10.1371/journal.pone.0086169')
## PeerJ
ft_get('10.7717/peerj.228')
## eLife
ft_get('10.7554/eLife.03032')
## BMC
ft_get(c('10.1186/2049-2618-2-7', '10.1186/2193-1801-3-7'))
## FrontiersIn
res <- ft_get(c('10.3389/fphar.2014.00109', '10.3389/feart.2015.00009'))
## Hindawi - via Entrez
res <- ft_get(c('10.1155/2014/292109','10.1155/2014/162024','10.1155/2014/249309'))
## F1000Research - via Entrez
ft_get('10.12688/f1000research.6522.1')
## Two different publishers via Entrez - retains publisher names
res <- ft_get(c('10.1155/2014/292109', '10.12688/f1000research.6522.1'))
res$hindawi
res$f1000research
## Pensoft
ft_get('10.3897/zookeys.499.8360')
### you'll need to specify the publisher for a DOI from a recent publication
ft_get('10.3897/zookeys.515.9332', from = "pensoft")
## Copernicus
out <- ft_get(c('10.5194/angeo-31-2157-2013', '10.5194/bg-12-4577-2015'))
```

```
out$copernicus
## arXiv - only pdf, you have to pass in the from parameter
res <- ft_get(x='cond-mat/9309029', from = "arxiv", cache=TRUE, backend="rds")
res %>% ft_extract
## bioRxiv - only pdf
res <- ft_get(x='10.1101/012476')
res$biorxiv
## Karger Publisher
ft_get('10.1159/000369331')
## CogentOA Publisher
ft_get('10.1080/23311916.2014.938430')
## MDPI Publisher
ft_get('10.3390/nu3010063')
ft_get('10.3390/nu7085279')
ft_get(c('10.3390/nu3010063', '10.3390/nu7085279')) # not working, only getting 1

# If you know the publisher, give DOI and publisher
## by default, PLOS gives back XML
ft_get('10.1371/journal.pone.0086169', from='plos')
## you can instead get json
ft_get('10.1371/journal.pone.0086169', from='plos', plosopts=list(wt="json"))

(dois <- searchplos(q="*:*", fl='id',
   fq=list('doc_type:full',"article_type:\"research article\""), limit=5)$data$id)
ft_get(dois, from='plos')
ft_get(c('10.7717/peerj.228','10.7717/peerj.234'), from='entrez')

# elife
ft_get('10.7554/eLife.04300', from='elife')
ft_get(c('10.7554/eLife.04300', '10.7554/eLife.03032'), from='elife')
## search for elife papers via Entrez
dois <- ft_search("elife[journal]", from = "entrez")
ft_get(dois)

# bmc
ft_get('http://www.microbiomejournal.com/content/download/xml/2049-2618-2-7.xml', from='bmc')
urls <- c('http://www.biomedcentral.com/content/download/xml/1471-2393-14-71.xml',
 'http://www.springerplus.com/content/download/xml/2193-1801-3-7.xml',
 'http://www.microbiomejournal.com/content/download/xml/2049-2618-2-7.xml')
ft_get(urls, from='bmc')

# Frontiers in Pharmacology (publisher: Frontiers)
doi <- '10.3389/fphar.2014.00109'
ft_get(doi, from="entrez")

# Hindawi Journals
ft_get(c('10.1155/2014/292109','10.1155/2014/162024','10.1155/2014/249309'), from='entrez')
res <- ft_search(query='ecology', from='crossref', limit=50,
                 crossrefopts = list(filter=list(has_full_text = TRUE,
                                                 member=98,
                                                 type='journal-article')))

out <- ft_get(res$crossref$data$DOI[1:20], from='entrez')
```

```
# Frontiers Publisher - Frontiers in Aging Nueroscience
res <- ft_get("10.3389/fnagi.2014.00130", from='entrez')
res$entrez

# Search entrez, get some DOIs
(res <- ft_search(query='ecology', from='entrez'))
res$entrez$data$doi
ft_get(res$entrez$data$doi[1], from='entrez')
ft_get(res$entrez$data$doi[1:3], from='entrez')

# Caching
res <- ft_get('10.1371/journal.pone.0086169', from='plos', cache=TRUE, backend="rds")

# Search entrez, and pass to ft_get()
(res <- ft_search(query='ecology', from='entrez'))
ft_get(res)

## End(Not run)
```

`ft_providers` *Search for information on journals or publishers.*

Description

Search for information on journals or publishers.

Usage

```
ft_providers(journal = NULL, publisher = NULL, limit = 10, ...)
```

Arguments

`journal`	Query terms
`publisher`	Source to query
`limit`	Number of records to return.
`...`	Further args passed on to `GET`

Value

An object of class ft_p

Examples

```
## Not run:
ft_providers(journal="Stem Cells International")
ft_providers(publisher="hindawi")
ft_providers(publisher="journal")

## End(Not run)
```

ft_search *Search for full text*

Description

ft_search is a one stop shop for searching for articles across many publishers and repositories. We currently support search for PLOS via the rplos package, Crossref via the rcrossref package, Entrez via the rentrez package, arXiv via the aRxiv package, and BMC and Biorxiv via internal helper functions in this package. Many publishers content is searchable via Crossref and Entrez - of course this doesn't mean that we can get full text for those articles. In the output objects of this function, we attempt to help by indicating what license is used for articles.

Usage

```
ft_search(query, from = "plos", limit = 10, plosopts = list(),
  bmcopts = list(), crossrefopts = list(), entrezopts = list(),
  arxivopts = list(), biorxivopts = list(), ...)
```

Arguments

query	Query terms
from	Source to query
limit	Number of records to return.
plosopts	PLOS options. See ?searchplos
bmcopts	BMC options. See ?bmc_search
crossrefopts	Crossref options. See ?cr_works
entrezopts	Entrez options. See ?entrez_search
arxivopts	arxiv options. See ?arxiv_search
biorxivopts	biorxiv options. See ?bx_search
...	Further args passed on to GET. Not working right now...

Value

An object of class ft, and objects of class ft_ind within each source

Examples

```
## Not run:
# Plos
(res1 <- ft_search(query='ecology', from='plos'))
res1$plos
ft_search(query='climate change', from='plos', limit=500, plosopts=list(
   fl=c('id','author','eissn','journal','counter_total_all','alm_twitterCount')))

# Crossref
```

```
(res2 <- ft_search(query='ecology', from='crossref'))
res2$crossref

# BioRxiv
(res <- ft_search(query='ecology', from='biorxiv'))
res$biorxiv

# Entrez
(res <- ft_search(query='ecology', from='entrez'))
res$entrez

# arXiv
(res <- ft_search(query='ecology', from='arxiv'))
res$arxiv

# BMC - can be very slow
(res <- ft_search(query='ecology', from='bmc'))
res$bmc

# PLOS, Crossref, and arxiv
(res <- ft_search(query='ecology', from=c('plos','crossref','arxiv')))
res$plos
res$arxiv
res$crossref

## End(Not run)
```

`ft_serialize` *Serialize raw text to other formats, including to disk*

Description

`ft_serialize` helps you convert to various data formats. If your data is in unparsed XML (i.e., character class), you can convert to parsed XML. If in XML, you can convert to (ugly-ish) JSON, or a list. In addition, this function allows you to save to various places, including Rds files, cached via `R.cache`, or to Redis.

Usage

```
ft_serialize(x, to = "xml", from = NULL, ...)

ft_get_keys(x)
```

Arguments

`x`	Input object, output from a call to `ft_get`. Required.
`to`	(character) Format to serialize to. One of list, xml, json, ... Required. Output to xml returns object of class XMLInternalDocument.

from	(character) Format x is currently in. Function attempts to use metadata provided, or guess from data itself. Optional. CURRENTLY IGNORED.
...	Further args passed on to read_xml or toJSON

Value

An object of class ft_parsed

Examples

```
## Not run:
dois <- c('10.1371/journal.pone.0087376','10.1371%2Fjournal.pone.0086169',
'10.1371/journal.pone.0102976','10.1371/journal.pone.0105225',
'10.1371/journal.pone.0102722','10.1371/journal.pone.0033693')
res <- ft_get(dois, from='plos')

# if articles in xml format, parse the XML
(out <- ft_serialize(res, to='xml'))
out$plos$data$data[[1]] # the xml

# From XML to JSON
(out <- ft_serialize(res, to='json'))
out$plos$data$data$`10.1371/journal.pone.0087376` # the json
jsonlite::fromJSON(out$plos$data$data$`10.1371/journal.pone.0087376`)

# To a list
out <- ft_serialize(res, to='list')
out$plos$data$data[[4]]
out$plos$data$data[[4]][[2]]$`article-meta`

# To various data stores on disk
## To an .Rds file
ft_serialize(res, to='file')

## To local files using R.cache package
res_rcache <- ft_serialize(res, to='rcache')

## To Redis
res_redis <- ft_serialize(res, to='redis')

# Chain together functions
doi <- '10.1371/journal.pone.0086169'
ft_get(doi, from='plos') %>%
   ft_serialize(to='xml') %>%
   ft_serialize(to='redis')

## End(Not run)
```

pdfx *PDF-to-XML conversion of scientific articles using pdfx*

Description

Uses a web service provided by Utopia at http://pdfx.cs.man.ac.uk/. Beware, this can be quite slow. pdfx posts the pdf from your machine to the web service, pdfx_html takes the output of pdfx and gives back a html version of extracted text, and pdfx_targz gives a tar.gz version of the extracted text.

Usage

```
pdfx(file, what = "parsed", ...)

pdfx_html(input, ...)

pdfx_targz(input, write_path, ...)
```

Arguments

file	(character) Path to a file, or files on your machine. Required.
what	(character) One of parsed or text.
...	Further args passed to GET. These aren't named, so just do e.g. , verbose(), or timeout(3)
input	Output from pdfx function
write_path	Path to write tar ball to.

Value

pdfx gives XML parsed to xml_document, pdfx_html gives html, pdfx_targz writes a tar.gz file to disk.

Examples

```
## Not run:
path <- system.file("examples", "example2.pdf", package = "fulltext")
pdfx(file = path)

out <- pdfx(file = path)
pdfx_html(out)

out <- pdfx(file = path)
tarfile <- tempfile(fileext = "tar.gz")
pdfx_targz(input = out, write_path = tarfile)

## End(Not run)
```

Index

∗Topic **package**
fulltext-package, 2

bmc_xml, *11*

cache, 3, *11*
cache_clear *(*cache*)*, 3
cache_options_get *(*cache*)*, 3
cache_options_set *(*cache*)*, 3
chunks, *2*, 4
collect, 6
Corpus, *10*

entrez_fetch, *11*
entrez_search, *11*

ft_browse, 7
ft_browse_sections *(*ft_browse*)*, 7
ft_extract, *2*, 8, *10*
ft_extract_corpus, 9
ft_get, *2*, *4*, *7*, *8*, 10
ft_get_keys *(*ft_serialize*)*, 16
ft_providers, 14
ft_search, *2*, *11*, *12*, 15
ft_serialize, *2*, 16
fulltext *(*fulltext-package*)*, 2
fulltext-package, 2

GET, *11*, *14*, *15*, *18*
get_text *(*collect*)*, 6

pdfx, 18
pdfx_html *(*pdfx*)*, 18
pdfx_targz *(*pdfx*)*, 18
plos_fulltext, *11*
print.gs_char *(*ft_extract*)*, 8
print.xpdf_char *(*ft_extract*)*, 8

R.cache, *16*
read_xml, *17*

tabularize *(*chunks*)*, 4
toJSON, *17*