Package 'oai'

May 13, 2021

Type Package

Title General Purpose 'Oai-PMH' Services Client

Description A general purpose client to work with any 'OAI-PMH' (Open Archives Initiative Protocol for 'Metadata' Harvesting) service. The 'OAI-PMH' protocol is described at <http://www.openarchives.org/OAI/openarchivesprotocol.html>. Functions are provided to work with the 'OAI-PMH' verbs: 'GetRecord', 'Identify', 'ListIdentifiers', 'ListMetadataFormats', 'ListRecords', and 'ListSets'.

Version 0.3.2

License MIT + file LICENSE

URL https://docs.ropensci.org/oai/, https://github.com/ropensci/oai

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

VignetteBuilder knitr

LazyData true

Encoding UTF-8

Imports xml2 (>= 1.0.0), httr (>= 1.2.0), plyr (>= 1.8.4), stringr (>= 1.1.0), tibble (>= 1.2)

Suggests DBI, knitr, RSQLite, testthat, markdown

RoxygenNote 7.1.1

NeedsCompilation no

Author Scott Chamberlain [aut, cre], Michal Bojanowski [aut] (supported by NCN grant 2012/07/D/HS6/01971)

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2021-05-13 21:20:07 UTC

R topics documented:

`oai-package` *OAI-PMH Client*

Description

oai is an R client to work with OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting) services, a protocol developed by the Open Archives Initiative (https://en.wikipedia.org/wiki/Open_Archives_Initiative). OAI-PMH uses XML data format transported over HTTP.

OAI-PMH Info

See the OAI-PMH V2 specification at `http://www.openarchives.org/OAI/openarchivesprotocol.html`

Implementation details

oai is built on **xml2** and **httr**. In addition, we give back data.frame's whenever possible to make data comprehension, manipulation, and visualization easier. We also have functions to fetch a large directory of OAI-PMH services - it isn't exhaustive, but does contain a lot.

Paging

Instead of paging with e.g., `page` and `per_page` parameters, OAI-PMH uses (optionally) `resumptionTokens`, with an optional expiration date. These tokens can be used to continue on to the next chunk of data, if the first request did not get to the end. Often, OAI-PMH services limit each request to 50 records, but this may vary by provider, I don't know for sure. The API of this package is such that we `while` loop for you internally until we get all records. We may in the future expose e.g., a `limit` parameter so you can say how many records you want, but we haven't done this yet.

Acknowledgements

Michal Bojanowski contributions were supported by (Polish) National Science Center (NCN) through grant 2012/07/D/HS6/01971.

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>

Michal Bojanowski <michal2992@gmail.com>

`count_identifiers` *Count OAI-PMH identifiers for a data provider.*

Description

Count OAI-PMH identifiers for a data provider.

Usage

```
count_identifiers(url = "http://export.arxiv.org/oai2", prefix = "oai_dc", ...)
```

Arguments

`url`	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
`prefix`	Specifies the metadata format that the records will be returned in
`...`	Curl options passed on to `GET`

Details

Note that some OAI providers do not include the entry `completeListSize` (http://www.openarchives.org/OAI/openarchivesprotocol.html#FlowControl) in which case we return an NA - which does not mean 0, but rather we don't know.

Examples

```
## Not run:
count_identifiers()

# curl options
# library("httr")
# count_identifiers(config = verbose())

## End(Not run)
```

dumpers *Result dumpers*

Description

Result dumpers are functions allowing to handle the chunks of results from OAI-PMH service "on the fly". Handling can include processing, writing to files, databases etc.

Usage

```
dump_raw_to_txt(
  res,
  args,
  as,
  file_pattern = "oaidump",
  file_dir = ".",
  file_ext = ".xml"
)

dump_to_rds(
  res,
  args,
  as,
  file_pattern = "oaidump",
  file_dir = ".",
  file_ext = ".rds"
)

dump_raw_to_db(res, args, as, dbcon, table_name, field_name, ...)
```

Arguments

`res`	results, depends on `as`, not to be specified by the user
`args`	list, query arguments, not to be specified by the user
`as`	character, type of result to return, not to be specified by the user
`file_pattern, file_dir, file_ext`	character respectively: initial part of the file name, directory name, and file extension used to create file names. These arguments are passed to `tempfile()` arguments `pattern`, `tmpdir`, and `fileext` respectively.
`dbcon`	**DBI**-compliant database connection
`table_name`	character, name of the database table to write into
`field_name`	character, name of the field in database table to write into
`...`	arguments passed to/from other functions

Details

Often the result of a request to a OAI-PMH service are so large that it is split into chunks that need to be requested separately using `resumptionToken`. By default functions like `list_identifiers()` or `list_records()` request these chunks under the hood and return all concatenated in a single R object. It is convenient but insufficient when dealing with large result sets that might not fit into RAM. A result dumper is a function that is called on each result chunk. Dumper functions can write chunks to files or databases, include initial pre-processing or extraction, and so on.

A result dumper needs to be function that accepts at least the arguments: `res`, `args`, `as`. They will get values by the enclosing function internally. There may be additional arguments, including `...`. Dumpers should return `NULL` or a value that will be collected and returned by the function calling the dumper (e.g. `list_records()`).

Currently result dumpers can be used with functions: `list_identifiers()`, `list_records()`, and `list_sets()`. To use a dumper with one of these functions you need to:

- Pass it as an additional argument `dumper`
- Pass optional addtional arguments to the dumper function in a list as the `dumper_args` argument

See Examples. Below we provide more details on the dumpers currently implemented.

`dump_raw_to_txt` writes raw XML to text files. It requires `as=="raw"`. File names are created using `tempfile()`. By default they are written in the current working directory and have a format `oaidump*.xml` where `*` is a random string in hex.

`dump_to_rds` saves results in an `.rds` file via `saveRDS()`. Type of object being saved is determined by the `as` argument. File names are generated in the same way as by `dump_raw_to_txt`, but with default extension `.rds`

`dump_xml_to_db` writes raw XML to a single text column of a table in a database. Requires `as == "raw"`. Database connection `dbcon` should be a connection object as created by `DBI::dbConnect()` from package **DBI**. As such, it can connect to any database supported by **DBI**. The records are written to a field `field_name` in a table `table_name` using `DBI::dbWriteTable()`. If the table does not exist, it is created. If it does, the records are appended. Any additional arguments are passed to `DBI::dbWriteTable()`

Value

Dumpers should return `NULL` or a value that will be collected and returned by the function using the dumper.

`dump_raw_to_txt` returns the name of the created file.

`dump_to_rds` returns the name of the created file.

`dump_xml_to_db` returns `NULL`

References

OAI-PMH specification https://www.openarchives.org/OAI/openarchivesprotocol.html

See Also

Functions supporting the dumpers: `list_identifiers()`, `list_sets()`, and `list_records()`

Examples

```
## Not run:

### Dumping raw XML to text files

# This will write a set of XML files to a temporary directory
fnames <- list_identifiers(from="2018-06-01T",
                           until="2018-06-14T",
                           as="raw",
                           dumper=dump_raw_to_txt,
                           dumper_args=list(file_dir=tempdir()))
# vector of file names created
str(fnames)
all( file.exists(fnames) )
# clean-up
unlink(fnames)


### Dumping raw XML to a database

# Connect to in-memory SQLite database
con <- DBI::dbConnect(RSQLite::SQLite(), dbname=":memory:")
# Harvest and dump the results into field "bar" of table "foo"
list_identifiers(from="2018-06-01T",
                 until="2018-06-14T",
                 as="raw",
                 dumper=dump_raw_to_db,
                 dumper_args=list(dbcon=con,
                                  table_name="foo",
                                  field_name="bar") )
# Count records, should be 101
DBI::dbGetQuery(con, "SELECT count(*) as no_records FROM foo")

DBI::dbDisconnect(con)




## End(Not run)
```

`get_records` *Get records*

Description

Get records

Usage

```
get_records(
  ids,
  prefix = "oai_dc",
  url = "http://api.gbif.org/v1/oai-pmh/registry",
  as = "parsed",
  ...
)
```

Arguments

ids	The OAI-PMH identifier for the record. One or more. Required.
prefix	specifies the metadata format that the records will be returned in. Default: oai_dc
url	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
as	(character) What to return. One of "parsed" (default), or "raw" (raw text)
...	Curl options passed on to GET

Details

There are some finite set of results based on the OAI prefix. We will provide parsers as we have time, and as users express interest. For prefix types we have parsers for we return a list of data.frame's, for each identifier, one data.frame for the header bits of data, and one data.frame for the metadata bits of data.

For prefixes we don't have parsers for, we fall back to returning raw XML, so you can at least parse the XML yourself.

Because some XML nodes are duplicated, we join values together of duplicated node names, separated by a semicolon (;) with no spaces. You can seprarate them yourself easily.

Value

a named list of data.frame's, or lists, or raw text

Examples

```
## Not run:
get_records("87832186-00ea-44dd-a6bf-c2896c4d09b4")

ids <- c("87832186-00ea-44dd-a6bf-c2896c4d09b4",
  "d981c07d-bc43-40a2-be1f-e786e25106ac")
(res <- get_records(ids))
lapply(res, "[[", "header")
lapply(res, "[[", "metadata")
do.call(rbind, lapply(res, "[[", "header"))
do.call(rbind, lapply(res, "[[", "metadata"))
```

```
# Get raw text
get_records("d981c07d-bc43-40a2-be1f-e786e25106ac", as = "raw")

# from arxiv.org
get_records("oai:arXiv.org:0704.0001", url = "http://export.arxiv.org/oai2")

## End(Not run)
```

`id` *Identify the OAI-PMH service for each data provider.*

Description

Identify the OAI-PMH service for each data provider.

Usage

```
id(url, as = "parsed", ...)
```

Arguments

`url`	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
`as`	(character) What to return. One of "parsed" (default), or "raw" (raw text)
`...`	Curl options passed on to `GET`

Examples

```
## Not run:
# arxiv
id("http://export.arxiv.org/oai2")

# GBIF - http://www.gbif.org/
id("http://api.gbif.org/v1/oai-pmh/registry")

# get back text instead of parsed
id("http://export.arxiv.org/oai2", as = "raw")
id("http://api.gbif.org/v1/oai-pmh/registry", as = "raw")

# curl options
library("httr")
id("http://export.arxiv.org/oai2", config = verbose())

## End(Not run)
```

`list_identifiers` *List OAI-PMH identifiers*

Description

List OAI-PMH identifiers

Usage

```
list_identifiers(
  url = "http://api.gbif.org/v1/oai-pmh/registry",
  prefix = "oai_dc",
  from = NULL,
  until = NULL,
  set = NULL,
  token = NULL,
  as = "df",
  ...
)
```

Arguments

`url`	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
`prefix`	Specifies the metadata format that the records will be returned in.
`from`	specifies that records returned must have been created/update/deleted on or after this date.
`until`	specifies that records returned must have been created/update/deleted on or before this date.
`set`	specifies the set that returned records must belong to.
`token`	a token previously provided by the server to resume a request where it last left off.
`as`	(character) What to return. One of "df" (for data.frame; default), "list", or "raw" (raw text)
`...`	Curl options passed on to `GET`

Examples

```
## Not run:
# from
recently <- format(Sys.Date() - 1, "%Y-%m-%d")
list_identifiers(from = recently)

# from and until
```

```
list_identifiers(from = '2018-06-01T', until = '2018-06-14T')

# set parameter - here, using ANDS - Australian National Data Service
list_identifiers(from = '2018-09-01T', until = '2018-09-05T',
  set = "dataset_type:CHECKLIST")

## End(Not run)
```

`list_metadataformats` *List available metadata formats from various providers.*

Description

List available metadata formats from various providers.

Usage

```
list_metadataformats(
  url = "http://api.gbif.org/v1/oai-pmh/registry",
  id = NULL,
  ...
)
```

Arguments

`url`	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
`id`	The OAI-PMH identifier for the record. Optional.
`...`	Curl options passed on to `GET`

Examples

```
## Not run:
list_metadataformats()

# no metadatformats for an identifier
list_metadataformats(id = "9da8a65a-1b9b-487c-a564-d184a91a2705")

# metadatformats available for an identifier
list_metadataformats(id = "ad7295e0-3261-4028-8308-b2047d51d408")

## End(Not run)
```

`list_records` *List records*

Description

List records

Usage

```
list_records(
  url = "http://api.gbif.org/v1/oai-pmh/registry",
  prefix = "oai_dc",
  from = NULL,
  until = NULL,
  set = NULL,
  token = NULL,
  as = "df",
  ...
)
```

Arguments

`url`	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
`prefix`	specifies the metadata format that the records will be returned in. Default: `oai_dc`
`from`	specifies that records returned must have been created/update/deleted on or after this date.
`until`	specifies that records returned must have been created/update/deleted on or before this date.
`set`	specifies the set that returned records must belong to.
`token`	(character) a token previously provided by the server to resume a request where it last left off. 50 is max number of records returned. We will loop for you internally to get all the records you asked for.
`as`	(character) What to return. One of "df" (for data.frame; default), "list", or "raw" (raw text)
`...`	Curl options passed on to `GET`

Examples

```
## Not run:
# By default you get back a single data.frame
list_records(from = '2018-05-01T00:00:00Z', until = '2018-05-03T00:00:00Z')
list_records(from = '2018-05-01T', until = '2018-05-04T')
```

```
# Get a list
list_records(from = '2018-05-01T', until = '2018-05-04T', as = "list")

# Get raw text
list_records(from = '2018-05-01T', until = '2018-05-04T', as = "raw")
list_records(from = '2018-05-01T', until = '2018-05-04T', as = "raw")

# Use a resumption token
# list_records(token =
#  "1443799900201,2015-09-01T00:00:00Z,2015-10-01T23:59:59Z,50,null,oai_dc")

## End(Not run)
```

`list_sets` *List sets*

Description

List sets

Usage

```
list_sets(
  url = "http://api.gbif.org/v1/oai-pmh/registry",
  token = NULL,
  as = "df",
  ...
)
```

Arguments

`url`	(character) OAI-PMH base url. Defaults to the URL for arXiv's OAI-PMH server (http://export.arxiv.org/oai2) or GBIF's OAI-PMH server (http://api.gbif.org/v1/oai-pmh/registry)
`token`	(character) a token previously provided by the server to resume a request where it last left off
`as`	(character) What to return. One of "df" (for data.frame; default), "list", or "raw" (raw text)
`...`	Curl options passed on to `GET`

Examples

```
## Not run:
# Get back a data.frame
list_sets()

# Get back a list
```

```
list_sets(as = "list")

# Get back raw text
list_sets(as = "raw")

# curl options
library("httr")
list_sets(config = verbose())

## End(Not run)
```

`load_providers` *Load an updated cache*

Description

Load an updated cache

Usage

```
load_providers(path = NULL, envir = .GlobalEnv)
```

Arguments

`path`	location where cache is located. Leaving to NULL loads the version in the installed package
`envir`	R environment to load data in to.

Details

Loads the data object providers into the global workspace.

Value

loads the object providers into the working space.

See Also

`update_providers()`

Examples

```
## Not run:
# By default the new providers table goes to directory ".", so just
# load from there
update_providers()
load_providers(path=".")

# Loads the version in the package
```

```
load_providers()

## End(Not run)
```

`oai_available` *Test of OAI-PMH service is available*

Description

Silently test if OAI-PMH service is available under the URL provided.

Usage

```
oai_available(u, ...)
```

Arguments

u	base URL to OAI-PMH service
...	other arguments passed to `id()`

Value

`TRUE` or `FALSE` if the service is available.

Examples

```
## Not run:
url_list <- list(
  archivesic="http://archivesic.ccsd.cnrs.fr/oai/oai.php",
  datacite = "http://oai.datacite.org/oai",

  # No OAI-PMH here
  google = "http://google.com"
)

sapply(url_list, oai_available)

## End(Not run)
```

providers *Metadata providers data.frame.*

Description

Metadata providers data.frame.

Value

A data.frame of three columns:

- repo_name - Name of the OAI repository
- base_url - Base URL of the OAI repository
- oai_identifier - OAI identifier for the OAI repository

update_providers *Update the locally stored OAI-PMH data providers table.*

Description

Data comes from http://www.openarchives.org/Register/BrowseSites. It includes the oai-identifier (if they have one) and the base URL. The website has the name of the data provider too, but not provided in the data pulled down here, but you can grab the name using the example below.

Usage

```
update_providers(path = ".", ...)
```

Arguments

path	Path to put data in.
...	Curl options passed on to httr::GET()

Details

This table is scraped from http://www.openarchives.org/Register/BrowseSites. I would get it from http://www.openarchives.org/pmh/registry/ListFriends, but it does not include repository names.

This function updates the table for you. Does take a while though, so go get a coffee.

See Also

load_providers()

Examples

```
## Not run:
update_providers()
load_providers()

## End(Not run)
```

Index