Package ‘finch’

October 13, 2022

Title Parse Darwin Core Files

Description Parse and create Darwin Core (<http://rs.tdwg.org/dwc/>) Simple and Archives. Functionality includes reading and parsing all the files in a Darwin Core Archive, including the datasets and metadata; read and parse simple Darwin Core files; and validation of Darwin Core Archives.

Version 0.4.0

License MIT + file LICENSE

LazyData true

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

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

Encoding UTF-8

Imports xml2 (>= 1.0.0), EML (>= 2.0.0), data.table (>= 1.10.0), digest, hoardr (>= 0.2.0)

Suggests testthat, crul, jsonlite

RoxygenNote 7.1.1

X-schema.org-applicationCategory Biology

X-schema.org-keywords biology, occurrences, specimen, biodiversity, collections, species

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>)

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

Repository CRAN

Date/Publication 2020-08-11 04:50:02 UTC

R topics documented:

`finch-package` *finch*

Description

Parse Darwin Core Archive files

Author(s)

Scott Chamberlain <`myrmecocystus@gmail.com`>

`as.location` *Convert a path or URL to a location object*

Description

Convert a path or URL to a location object

Usage

```
as.location(x, ...)

## S3 method for class 'character'
as.location(x, ...)

## S3 method for class 'location'
as.location(x, ...)

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

Arguments

`x`	Input, a path or URL
`...`	Ignored.

Examples

```
# A zip file
file <- system.file("examples/0000154-150116162929234.zip",
  package = "finch")
as.location(file)

# A directory
dir <- system.file("examples/0000154-150116162929234",
  package = "finch")
as.location(dir)

# A URL
as.location("https://httpbin.org/get")
```

---

dwca_cache *Caching*

Description

Manage cached `finch` files with package **hoardr**

Details

The dafault cache directory is `paste0(rappdirs::user_cache_dir(), "/R/finch")`, but you can set your own path using `cache_path_set()`

`cache_delete` only accepts one file name, while `cache_delete_all` doesn't accept any names, but deletes all files. For deleting many specific files, use `cache_delete` in a `lapply()` type call

Useful user functions

- `dwca_cache$cache_path_get()` get cache path
- `dwca_cache$cache_path_set()` set cache path
- `dwca_cache$list()` returns a character vector of full path file names
- `dwca_cache$files()` returns file objects with metadata
- `dwca_cache$details()` returns files with details
- `dwca_cache$delete()` delete specific files
- `dwca_cache$delete_all()` delete all files, returns nothing

Examples

```
## Not run:
dwca_cache

# list files in cache
dwca_cache$list()
```

```
# delete certain database files
# dwca_cache$delete("file path")
# dwca_cache$list()

# delete all files in cache
# dwca_cache$delete_all()
# dwca_cache$list()

# set a different cache path from the default

## End(Not run)
```

---

dwca_read *Parse Darwin Core Archive*

Description

Parse Darwin Core Archive

Usage

```
dwca_read(input, read = FALSE, ...)
```

Arguments

`input`	(character) Path to local zip file, directory, or a url. If a URL it must be for a zip file.
`read`	(logical) Whether or not to read in data files. If `FALSE`, we give back paths to files only. Default: `FALSE`
`...`	Further args passed on to `data.table::fread()`

Details

Note that sometimes file reads fail. We use `data.table::fread()` internally, which is very fast, but can fail sometimes. If so, try reading in the data manually.

When you pass in a URL, we use **rappdirs** to determine cache path, and if you pass the same URL again, and your cache is not cleared, we'll pull from the cache. Passing a file or directory on your local system won't invoke the caching route, but will go directly to the file/directory.

Examples

```
## Not run:
# set up a temporary directory for the example
dwca_cache$cache_path_set(path = "finch", type = "tempdir")

dir <- system.file("examples", "0000154-150116162929234", package = "finch")

# Don't read data in
```

```
(x <- dwca_read(dir, read=FALSE))
x$files
x$highmeta
x$dataset_meta[[1]]
x$data

# Read data
(x <- dwca_read(dir, read=TRUE))
head(x$data[[1]])

# Can pass in a zip file
zip <- system.file("examples", "0000154-150116162929234.zip",
  package = "finch")
(out <- dwca_read(zip))
out$files
out$highmeta
out$emlmeta
out$dataset_meta

# Can pass in zip file as a url
url <-
"https://github.com/ropensci/finch/blob/master/inst/examples/0000154-150116162929234.zip?raw=true"
(out <- dwca_read(url))

# another url
url <- "http://ipt.jbrj.gov.br/jbrj/archive.do?r=redlist_2013_taxons&v=3.12"
(out <- dwca_read(url))

## End(Not run)
```

dwca_validate *Validate a Darwin Core Archive*

Description

Validate a Darwin Core Archive

Usage

```
dwca_validate(x, ifModifiedSince = NULL, browse = FALSE, ...)
```

Arguments

x	(character) A url for a Darwin Core Archive. If you have a local Darwin Core Archive, put it up online somewhere. Required.
ifModifiedSince	(character) An optional ISO date (yyyy-mm-dd) to enable conditional get requests, validating archives only if they have been modified since the given date. This feature requires the archive url to honor the if-modified-since http header.

	Apache webservers for example do this out of the box for static files, but if you use dynamic scripts to generate the archive on the fly this might not be recognised. Optional.
`browse`	(logical) Browse to generated report or not. Default: `FALSE`
`...`	Curl options passed to crul::HttpClient

Details

Uses the GBIF DCA validator (http://tools.gbif.org/dwca-validator/)

Examples

```
## Not run:
x <- "http://rs.gbif.org/datasets/german_sl.zip"
dwca_validate(x)

## End(Not run)
```

`finch-defunct` *Defunct functions in finch*

Description

- `dwca_cache_delete`: Defunt - see dwca_cache
- `dwca_cache_delete_all`: Defunt - see dwca_cache
- `dwca_cache_details`: Defunt - see dwca_cache
- `dwca_cache_list`: Defunt - see dwca_cache

`simple_read` *Parse a DarwinRecordSet and SimpleDarwinRecordSet files*

Description

Parse a DarwinRecordSet and SimpleDarwinRecordSet files

Usage

```
simple_read(file)
```

Arguments

`file`	(character) A path to a single simple Darwin Core file in XML format. Required.

Details

Make sure when reading a DarwinRecordSet to access the chunks by position rather than name since duplicate names are allowed in chunks.

Value

a S3 class `dwc_recordset` when a DarwinRecordSet is given, or a `dwc_simplerecordset` when a SimpleDarwinRecordSet is given. In each case the object is really just a list, with lightweight S3 class attached for easy downstream usage. Prints summary to screen by default

Examples

```
## Not run:
# SimpleDarwinRecordSet examples
file <- system.file("examples", "example_simple.xml", package = "finch")
simple_read(file)
file <- system.file("examples", "example_simple_fossil.xml",
  package = "finch")
simple_read(file)

# DarwinRecordSet examples
file <- system.file("examples", "example_classes_observation.xml",
  package = "finch")
simple_read(file)

file <- system.file("examples", "example_classes_specimen.xml",
  package = "finch")
simple_read(file)

# access elements of the object
file <- system.file("examples", "example_classes_specimen.xml",
  package = "finch")
res <- simple_read(file)
## namespaces
res$meta
## locations
res$locations
## chunks, the first one
res$chunks[[1]]

## End(Not run)
```

Index