

# Package ‘finch’

December 23, 2016

**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** 0.1.0

**License** MIT + file LICENSE

**LazyData** true

**URL** <https://github.com/ropensci/finch>

**BugReports** <https://github.com/ropensci/finch/issues>

**Imports** xml2 (>= 1.0.0), EML (>= 1.0.1), data.table (>= 1.10.0), plyr, rappdirs, digest

**Suggests** testthat, httr, jsonlite, covr

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Author** Scott Chamberlain [aut, cre]

**Maintainer** Scott Chamberlain <[myrmecocystus@gmail.com](mailto:myrmecocystus@gmail.com)>

**Repository** CRAN

**Date/Publication** 2016-12-23 18:18:19

## R topics documented:

finch-package . . . . .	2
as.location . . . . .	2
dwca_cache . . . . .	3
dwca_read . . . . .	4
dwca_validate . . . . .	5
simple_read . . . . .	6

<b>Index</b>	<b>8</b>
--------------	----------

finch-package            *Parse Darwin Core Archive files*

---

**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	<i>Manage cached files</i>
------------	----------------------------

---

## Description

Manage cached files

## Usage

```
dwca_cache_list()

dwca_cache_delete(files, force = TRUE)

dwca_cache_delete_all(force = TRUE)

dwca_cache_details(files = NULL)
```

## Arguments

files	(character) one or more complete file names
force	(logical) Should files be force deleted? Default: TRUE

## Details

cache\_delete only accepts 1 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

We cache using [user\\_cache\\_dir](#), find your cache folder by executing `rappdirs::user_cache_dir("finch-cache")`

## Functions

- `dwca_cache_list()` returns a character vector of full path file names
- `dwca_cache_delete()` deletes one or more files, returns nothing
- `dwca_cache_delete_all()` delete all files, returns nothing
- `dwca_cache_details()` prints file name and file size for each file, supply with one or more files, or no files (and get details for all available)

## Examples

```
## Not run:
# list files in cache
dwca_cache_list()

# List info for single files
dwca_cache_details(files = dwca_cache_list()[1])
dwca_cache_details(files = dwca_cache_list()[2])

# List info for all files
dwca_cache_details()

# delete files by name in cache
# dwca_cache_delete(files = dwca_cache_list()[1])

# delete all files in cache
# dwca_cache_delete_all()

## 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 <a href="#">fread</a>

## Details

Note that sometimes file reads fail. We use [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**

```

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

## Not run:
# 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 <a href="#">GET</a>

**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)
```

---

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**

```
# 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]]
```

# Index

## \*Topic **package**

finch-package, 2

as.location, 2

dwca\_cache, 3

dwca\_cache\_delete (dwca\_cache), 3

dwca\_cache\_delete\_all (dwca\_cache), 3

dwca\_cache\_details (dwca\_cache), 3

dwca\_cache\_list (dwca\_cache), 3

dwca\_read, 4

dwca\_validate, 5

finch (finch-package), 2

finch-package, 2

fread, 4

GET, 6

lapply, 3

print.location (as.location), 2

simple\_read, 6

user\_cache\_dir, 3