Package 'galah'

August 21, 2021

Type Package

Title Atlas of Living Australia (ALA) Data and Resources in R

Version 1.3.1

Description The Atlas of Living Australia ('ALA') provides tools to enable users of biodiversity information to find, access, combine and visualise data on Australian plants and animals; these have been made available from <https://ala.org.au/>. 'galah' provides a subset of the tools to be directly used within R. It enables the R community to directly access data and resources hosted by the 'ALA'.

Depends R (>= 4.0.0)

Imports assertthat, crul, data.table, data.tree, digest, httr, jsonlite (>= 0.9.8), sf, sp, stringr (>= 1.0.0), utils, wellknown

Suggests vcr (>= 0.6.0), collapsibleTree, covr, dplyr, ggplot2, knitr, magrittr, pkgdown, rmarkdown, taxize, testthat, treemapify, viridis

License MPL-2.0

URL https://github.com/AtlasOfLivingAustralia/galah

BugReports https://github.com/AtlasOfLivingAustralia/galah/issues

Maintainer Martin Westgate <martin.westgate@csiro.au>

LazyLoad yes

VignetteBuilder knitr

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

Author Martin Westgate [aut, cre],
Matilda Stevenson [aut],
Dax Kellie [aut],
Peggy Newman [aut]

Repository CRAN

Date/Publication 2021-08-21 09:20:02 UTC

R topics documented:

`ala_citation` *Generate a citation for occurrence data*

Description

If a `data.frame` was generated using `ala_occurrences`, and the `mint_doi` argument was set to `TRUE`, the DOI associated with that dataset is appended to the resulting `data.frame` as an attribute. This function simply formats that DOI as a citation that can be included in a scientific publication. Please also consider citing this package, using the information in `citation("galah")`.

Usage

```
ala_citation(data)
```

Arguments

`data`	data.frame: occurrence data generated by `ala_occurrences`

Value

A string containing the citation for that dataset.

`ala_config` *Get or set configuration options that control galah behaviour*

Description

The `galah` package supports large data downloads, and also interfaces with the ALA which requires that users of some services provide a registered email address and reason for downloading data. The `ala_config` function provides a way to manage these issues as simply as possible.

Usage

```
ala_config(..., profile_path = NULL)
```

Arguments

`...`	Options can be defined using the form `name = value`. Valid arguments are: • `atlas` string: Living Atlas to point to, Australia by default • `caching` logical: if TRUE, results will be cached, and any cached results will be re-used). If FALSE, data will be downloaded. • `cache_directory` string: the directory to use for the cache. By default this is a temporary directory, which means that results will only be cached within an R session and cleared automatically when the user exits R. The user may wish to set this to a non-temporary directory for caching across sessions. The directory must exist on the file system. • `download_reason_id` numeric or string: the "download reason" required. by some ALA services, either as a numeric ID (currently 0–11) or a string (see `find_reasons()` for a list of valid ID codes and names). By default this is NA. Some ALA services require a valid download_reason_id code, either specified here or directly to the associated R function. • `email` string: An email address that has been registered with ALA at this address. A registered email is required for some functions in `galah`. • `send_email` logical: should you receive an email for each query to `ala_occurrences()`? Defaults to `FALSE`; but can be useful in some instances, for example for tracking DOIs assigned to specific downloads for later citation. • `verbose` logical: should `galah` give verbose output to assist debugging? Defaults to FALSE. • `run_checks` logical: should `galah` run checks for filters and columns. If making lots of requests sequentially, checks can slow down the process and lead to HTTP 500 errors, so should be turned off. Defaults to TRUE.
`profile_path`	string: (optional), path to a directory to store config values in. If provided, config values will be written to a new or existing .Rprofile file for future sessions. `NULL` by default.

Value

For `ala_config()`, a `list` of all options. When `ala_config(...)` is called with arguments, nothing is returned but the configuration is set.

See Also

As of galah v1.3.0 please use `galah_config()` instead of `ala_config`.

Examples

```
## Not run:
 ala_config()
 ala_config(caching = FALSE)
 find_reasons()
 ala_config(download_reason_id = 0,verbose = TRUE)

## End(Not run)
```

`ala_counts` *Count of ALA records*

Description

Prior to downloading data it is often valuable to have some estimate of how many records are available, both for deciding if the query is feasible, and for estimating how long it will take to download. Alternatively, for some kinds of reporting, the count of observations may be all that is required, for example for understanding how observations are growing or shrinking in particular locations of for particular taxa. To this end, `ala_counts()` takes arguments in the same format as `ala_occurrences()`, and provides either a total count of records matching the criteria, or a `data.frame` of counts matching the criteria supplied to the `group_by` argument.

Usage

```
ala_counts(
  taxa = NULL,
  filters = NULL,
  locations = NULL,
  group_by,
  limit = 100,
  type = c("record", "species"),
  refresh_cache = FALSE
)
```

Arguments

`taxa`	`data.frame`: generated by a call to `select_taxa()`. This argument also accepts a vector of unique species identifiers.
`filters`	`data.frame`: generated by a call to `select_filters()`
`locations`	`string`: generated by a call to `select_locations()`
`group_by`	`string`: field to count by

limit	numeric: maximum number of categories to return, defaulting to 100. If limit is NULL, all results are returned. For some categories this will take a while.
type	string: one of c("record","species"). Defaults to "record". If "species", the number of species matching the criteria will be returned, if "record", the number of records matching the criteria will be returned.
refresh_cache	logical: if set to 'TRUE' and 'galah_config(caching = TRUE)' then files cached from a previous query will be replaced by the current query

Value

- A single count, if group_by is not specified or,
- A data.frame of counts by group_by field, if it is specified

Examples

```
## Not run:
# With no arguments, return the total number of records in the ALA
ala_counts()

# Group counts by state and territory
ala_counts(group_by = "stateProvince")

# Count records matching a filter
ala_counts(filters = select_filters(basisOfRecord = "FossilSpecimen"))

# Count the number of species recorded for each kingdom
ala_counts(group_by = "kingdom", type = "species")

## End(Not run)
```

---

ala_media *Images, sounds and videos*

Description

In addition to text data describing individual occurrences and their attributes, ALA stores images, sounds and videos associated with a given record. ala_media allows download of any and all of the media types.

Usage

```
ala_media(
  taxa = NULL,
  filters = NULL,
  locations = NULL,
  columns = select_columns(group = "basic"),
  download_dir,
  refresh_cache = FALSE
)
```

Arguments

taxa	data.frame: generated by a call to select_taxa(). This argument also accepts a vector of unique species identifiers.
filters	data.frame: generated by a call to select_filters()
locations	string: generated by a call to select_locations()
columns	data.frame: generated by a call to select_columns()
download_dir	string: path to directory to store the downloaded media in
refresh_cache	logical: if set to 'TRUE' and 'galah_config(caching = TRUE)' then files cached from a previous query will be replaced by the current query

Details

ala_occurrences() works by first finding all occurrence records matching the filters which contain media, then downloading the metadata for the media and the media files. select_filters() can take both filters relating to occurrences (e.g. basis of records), and filters relating to media (e.g. type of licence). It may be beneficial when requesting a large number of records to show a progress bar by setting verbose = TRUE in galah_config().

Value

data.frame of metadata of the downloaded media

See Also

ala_counts to find the number of records with media- note this is not necessarily the same as the number of media files, as each record can have more than one media file associated with it (see examples section for how to do this).

Examples

```
## Not run:
# Download Regent Honeyeater multimedia
media_data <- ala_media(
    taxa = select_taxa("Regent Honeyeater"),
    filters = select_filters(year = 2011),
    download_dir = "media")

# Specify a single media type to download
media_data <- ala_media(
     taxa = select_taxa("Eolophus Roseicapilla"),
     filters = select_filters(multimedia = "Sound"))

# Filter to only records with a particular licence type
media_data <- ala_media(
      taxa = select_taxa("Ornithorhynchus anatinus"),
      filters = select_filters(year = 2020,
      license = "http://creativecommons.org/licenses/by-nc/4.0/")
)
# Check how many records have media files
```

```
ala_counts(
     filters = select_filters(multimedia = c("Image","Sound","Video")),
     group_by = "multimedia"
)

## End(Not run)
```

`ala_occurrences` *Occurrence records*

Description

The most common form of data stored by ALA are observations of individual life forms, known as 'occurrences'. This function allows the user to search for occurrence records that match their specific criteria, and return them as a `data.frame` for analysis. Optionally, the user can also request a DOI for a given download to facilitate citation and re-use of specific data resources.

Usage

```
ala_occurrences(
  taxa = NULL,
  filters = NULL,
  locations = NULL,
  columns = select_columns(group = "basic"),
  mint_doi = FALSE,
  doi,
  refresh_cache = FALSE
)
```

Arguments

`taxa`	`data.frame`: generated by a call to `select_taxa()`. This argument also accepts a vector of unique species identifiers.
`filters`	`data.frame`: generated by a call to `select_filters()`
`locations`	`string`: generated by a call to `select_locations()`
`columns`	`data.frame`: generated by a call to `select_columns()`
`mint_doi`	`logical`: by default no DOI will be generated. Set to 'TRUE' if you intend to use the data in a publication or similar
`doi`	`string`: this argument enables retrieval of occurrence records previously downloaded from the ALA, using the DOI generated by the data.
`refresh_cache`	`logical`: if set to 'TRUE' and 'galah_config(caching = TRUE)' then files cached from a previous query will be replaced by the current query

Details

Note that unless care is taken, some queries can be particularly large. While most cases this will simply take a long time to process, if the number of requested records is >50 million the call will not return any data. Users can test whether this threshold will be reached by first calling `ala_counts()` using the same arguments that they intend to pass to `ala_occurrences()`. It may also be beneficial when requesting a large number of records to show a progress bar by setting `verbose = TRUE` in `galah_config()`.

Value

A `data.frame` of occurrences, columns as specified by `select_columns()`. The `data.frame` object has the following attributes:

- a listing of the user-supplied arguments of the `data_request` (i.e., taxa, filters, locations, columns)
- a `doi` of the data download
- the `search_url` of the query to ALA API

Examples

```
## Not run:
# Search for occurrences matching a taxon identifier
occ <- ala_occurrences(taxa = select_taxa("Reptilia"))

# Search for occurrences in a year range
occ <- ala_occurrences(filters = select_filters(year = seq(2010, 2020)))

# Search for occurrences in a WKT-specified area
polygon <- "POLYGON((146.24960 -34.05930,146.37045 -34.05930,146.37045 \
-34.152549,146.24960 -34.15254,146.24960 -34.05930))"
occ <- ala_occurrences(locations = select_locations(polygon))

## End(Not run)
```

---

ala_species *Species lists*

Description

While there are reasons why users may need to check every record meeting their search criteria (i.e. using `ala_occurrences`), a common use case is to simply identify which species occur in a specified region, time period, or taxonomic group. This function returns a `data.frame` with one row per species, and columns giving associated taxonomic information.

Usage

```
ala_species(
  taxa = NULL,
  filters = NULL,
  locations = NULL,
  refresh_cache = FALSE
)
```

Arguments

taxa	data.frame: generated by a call to select_taxa(). This argument also accepts a vector of unique species identifiers.
filters	data.frame: generated by a call to select_filters()
locations	string: generated by a call to select_locations()
refresh_cache	logical: if set to 'TRUE' and 'galah_config(caching = TRUE)' then files cached from a previous query will be replaced by the current query

Details

The primary use case of this function is to extract species-level information given a set of criteria defined by select_taxa(), select_filters() or select_locations(). If the purpose is simply to get taxonomic information that is not restricted by filtering, then select_taxa() is more efficient. Similarly, if counts are required that include filters but without returning taxonomic detail, then ala_counts() is more efficient (see examples).

Value

A data.frame of matching species. The data.frame object has attributes listing of the user-supplied arguments of the data_request (i.e., taxa, filters, locations, columns)

Examples

```
## Not run:

# Lookup genus "Heleioporus" in the ALA
select_taxa("Heleioporus")

# How many records are there for this genus?
ala_counts(select_taxa("Heleioporus"))
# or equivalently:
select_taxa("Heleioporus", counts = TRUE)

# How best to get taxonomic info on species within this genus?
# also includes a row for genus (i.e. not just species)
select_taxa("Heleioporus", children = TRUE)
# returns counts by species, but no taxonomic information
ala_counts(select_taxa("Heleioporus"), group_by = "species")
# every row is a species with associated taxonomic data
ala_species(select_taxa("Heleioporus"))
```

```
## End(Not run)
```

`ala_taxonomy` *Search taxonomic trees*

Description

The ALA has its' own internal taxonomy that is derived from authoritative sources. `search_taxonomy` provides a means to query that taxonomy, returning a tree (class `Node`) showing which lower clades are contained within the specified taxon.

Usage

```
ala_taxonomy(taxa, down_to)
```

Arguments

`taxa`	The identity of the clade for which a taxonomic hierarchy should be returned. Should be specified using an object of class `data.frame` and `ala_id`, as returned from `select_taxa()`.
`down_to`	`string`: A taxonomic rank to search down to. See `find_ranks()` for valid inputs.

Details

The approach used by this function is recursive, meaning that it becomes slow for large queries such as `search_taxonomy(select_taxa("Plantae"),down_to = "species")`. Although the inputs to `select_taxa` and `down_to` are case-insensitive, node names are always returned in title case.

Value

A tree consisting of objects of class `Node`, containing the requested taxonomy. Each node contains the following attributes:

- `name`: The scientific name of the taxon in question
- `rank`: The taxonomic rank to which that taxon belongs
- `guid`: A unique identifier used by the ALA
- `authority`: The source of the taxonomic name & identifier

See Also

`select_taxa` to search for an individual taxon; `find_ranks` for valid ranks used to specify the `down_to` argument.

Examples

```
## Not run:
search_taxonomy(select_taxa("chordata"), down_to = "class")

## End(Not run)
```

---

clear_cached_files *Clear previously cached files*

Description

Deletes cached files within the cached file directory and their query metadata

Usage

```
clear_cached_files()
```

Examples

```
## Not run:
## configure caching and create a query to cache
galah_config(caching = TRUE)
dat <- ala_counts(group_by = "year")

## clear cached files directory
clear_cached_files()

## End(Not run)
```

---

exclude *Negate a filter value*

Description

Deprecated alternative to `select_filters(field != value)`.

Usage

```
exclude(value)
```

Arguments

value	string: filter value(s) to be excluded

Value

value with class "exclude"

See Also

exclude is used with select_filters or select_taxa to exclude values

---

`find_atlases` *List supported Living Atlases*

Description

galah supports downloading data from a number of International Living Atlases. Use this function to get a list of all currently supported atlases.

Usage

```
find_atlases()
```

Value

a `data.frame` of Living Atlas information, including taxonomy source and information for each atlas.

See Also

This function is helpful in setting up `galah_config()`.

`find_cached_files` *List previously cached files*

Description

Uses query metadata stored in metadata.rds in the cache directory

Usage

```
find_cached_files()
```

Value

a `list` of available cached files, the function used to create them, and the filter object

Examples

```
## configure caching and create a query to cache
## Not run:
galah_config(caching = TRUE)
dat <- ala_counts(group_by = "year")

## list cached files
find_cached_files()

## End(Not run)
```

`find_field_values` *List valid options for a categorical field*

Description

When building a set of filters with `select_filters`, a user can use this function to check that the values provided are valid options.

Usage

```
find_field_values(field, limit = 20)
```

Arguments

`field`	`string`: field to return the categories for. Use `search_fields` to view valid fields.
`limit`	`numeric`: maximum number of categories to return. 20 by default.

Value

A `data.frame` containing columns `field` (user-supplied) and `category` (i.e. field values).

See Also

See `search_fields` for ways to use information returned by this function.

Examples

```
## Not run:
find_field_values("basisOfRecord")
find_field_values("stateProvince")

## End(Not run)
```

find_profiles *Data quality profiles*

Description

The ALA provides a number of pre-built data quality profiles for filtering data according to quality checks. A data quality profile can be specified in the `profile` argument in `select_filters()` and used to filter searches in `ala_occurrences()`, `ala_counts()` and `ala_species()`.

Usage

```
find_profiles()
```

Value

A `data.frame` of available profiles

See Also

This function gives viable profle names for passing to `select_filters()`. For more detail on a given profile see `find_profile_attributes()`.

Examples

```
## Not run:
# Get available profiles
profile_df <- find_profiles()
# Values given in the 'shortName' column are accepted by select_filter(), i.e.
select_filters(profile = profile_df$shortName[1])
# is equivalent to:
select_filters(profile = "ALA")

## End(Not run)
```

find_profile_attributes *Get data filters for a specified data quality profile*

Description

Each data quality profile is made up of a series of filters. While some users may wish to simply trust the default filters, it is often useful to check what information they return, particularly if advanced customization is needed. This function gives all of the arguments built into a specific profile.

Usage

```
find_profile_attributes(profile)
```

Arguments

profile	string: a data quality profile name, short name or id. See find_profiles() for valid filters

Value

A data.frame of profile attributes, consisting of a free text description and the actual filter used.

See Also

find_profiles() for a list of valid profiles; select_filters() for how to include this information in a data query.

Examples

```
profile_info <- find_profile_attributes("CSDM")
profile_info$description # free-text description of each filter in the "CSDM" profile
```

---

find_ranks *Find valid taxonomic ranks*

Description

Return taxonomic ranks recognised by the ALA.

Usage

```
find_ranks()
```

Value

A data.frame of available ranks

See Also

This function provides a reference that is useful when specifying the down_to argument of ala_taxonomy.

Examples

```
## Not run:
rank_df <- find_ranks()

## End(Not run)
```

find_reasons *List valid download reasons*

Description

When downloading occurrence data with ala_occurrences the ALA APIs require a reason for download to be specified. By default, a download reason of 'scientific research' is set for you, but if you wish to change this you can do so with galah_config(). Use this function to view the list of download reason code and names. When specifying a reason, you can use either the download code or name.

Usage

```
find_reasons()
```

Value

A data.frame of valid download reasons, containing the id and name for each reason.

See Also

This function is helpful in setting up galah_config().

galah **galah**

Description

galah is an R interface to the Atlas of Living Australia (ALA; https://www.ala.org.au/), a biodiversity data repository focussed primarily on observations of individual life forms. It also supports access to some other 'living atlases' that use the same computational infrastructure. The basic unit of data at ALA is an **occurrence** record, based on the 'Darwin Core' data standard (https://dwc.tdwg.org). galah enables users to locate and download species observations, taxonomic information, or associated media such images or sounds, and to restrict their queries to particular taxa or locations. Users can specify which columns are returned by a query, or restrict their results to observations that meet particular quality-control criteria.

Functions

Data

- ala_counts Count the number of records or species returned by a query
- ala_taxonomy Return a section of the ALA taxonomic tree
- ala_species Download species lists
- ala_occurrences Download occurrence records

- ala_media Download images and sounds

Filter

- select_taxa Search for taxonomic identifiers
- select_filters Filter records
- select_locations Specify a location
- select_columns Columns to return in an occurrence download

Lookup

- search_fields Free-text search for layers and fields
- find_field_values List possible values for a given field
- find_profiles List data quality profiles
- find_profile_attributes List filters included in a data quality profile
- find_ranks List available taxonomic ranks
- find_reasons List valid download reasons
- find_atlases List supported international atlases

Cache management

- find_cached_files List previously cached files and their metadata
- clear_cached_files Clear previously cached files and their metadata

Help

- galah_config Package configuration options
- ala_citation Citation for a dataset

Terminology

To get the most value from galah, it is helpful to understand some terminology used by the ALA. Each occurrence record contains taxonomic information, and usually some information about the observation itself, such as its location. In addition to this record-specific information, ALA appends contextual information to each record, particularly data from spatial **layers** reflecting climate gradients or political boundaries. ALA also runs a number of quality checks against each record, resulting in **assertions** attached to the record. Each piece of information associated with a given occurrence record is stored in a **field**, which corresponds to a **column** when imported to an R data.frame. See search_fields to view valid fields, layers and assertions.

Data fields are important because they provide a means to **filter** occurrence records; i.e. to return only the information that you need, and no more. Consequently, much of the architecture of galah has been designed to make filtering as simple as possible, by using functions with the select_ prefix. Each select function allows the user to filter in a different way, and again the function suffix contains this information. For example, you can choose which taxonomic groups are included using select_taxa(), or a specific location using select_locations(). By combining different filter functions, it is possible to build complex queries to return only the most valuable information for a given problem.

A notable extention of the filtering approach is to remove records with low 'quality'. ALA performs quality control checks on all records that it stores. These checks are used to generate new fields, that can then be used to filter out records that are unsuitable for particular applications. However, there are many possible data quality checks, and it is not always clear which are most appropriate in a given instance. Therefore, galah supports ALA data quality **profiles**, which can be passed to select_filters() to quickly remove undesirable records. A full list of data quality profiles is returned by find_profiles().

For those outside Australia, 'galah' is the common name of *Eolophus roseicapilla*, a widely-distributed Australian bird species.

Package design

In most cases, users will be primarily interested in using galah to return data from one of the living atlases. These functions are named with the prefix ala_, followed by a suffix describing the information that they provide. For example, we anticipate that users will wish to download occurrence data, which can be achieved using the function ala_occurrences(). However, it is also possible to download data on species via ala_species(), or media content (largely images) via ala_media(). Alternatively, users can assess how many records meet their particular criteria using ala_counts(). All functions return a data.frame as their standard format, except ala_taxonomy() which returns a data.tree.

Functions in galah are designed according to a nested architecture. Users that require data should begin by locating the relevant ala_ function; the arguments within that function then call correspondingly-named select_ functions; and finally the specific values that can be interpreted by those select_ functions are given by functions with the prefix search_ or find_. So, to limit occurrence downloads to a specific taxonomic group, for example, you pass the reult of select_taxa() to the taxa argument of ala_occurrences().

References

For more information on the ALA API, visit https://api.ala.org.au/. If you have any questions, comments or suggestions, please email support@ala.org.au.

---

galah_config *Get or set configuration options that control galah behaviour*

Description

The galah package supports large data downloads, and also interfaces with the ALA which requires that users of some services provide a registered email address and reason for downloading data. The galah_config function provides a way to manage these issues as simply as possible.

Usage

```
galah_config(..., profile_path = NULL)
```

Arguments

...	Options can be defined using the form `name = value`. Valid arguments are: • `atlas` string: Living Atlas to point to, Australia by default • `caching` logical: if TRUE, results will be cached, and any cached results will be re-used). If FALSE, data will be downloaded. • `cache_directory` string: the directory to use for the cache. By default this is a temporary directory, which means that results will only be cached within an R session and cleared automatically when the user exits R. The user may wish to set this to a non-temporary directory for caching across sessions. The directory must exist on the file system. • `download_reason_id` numeric or string: the "download reason" required. by some ALA services, either as a numeric ID (currently 0–11) or a string (see `find_reasons()` for a list of valid ID codes and names). By default this is NA. Some ALA services require a valid download_reason_id code, either specified here or directly to the associated R function. • `email` string: An email address that has been registered with ALA at this address. A registered email is required for some functions in `galah`. • `send_email` logical: should you receive an email for each query to `ala_occurrences()`? Defaults to `FALSE`; but can be useful in some instances, for example for tracking DOIs assigned to specific downloads for later citation. • `verbose` logical: should `galah` give verbose output to assist debugging? Defaults to FALSE. • `run_checks` logical: should `galah` run checks for filters and columns. If making lots of requests sequentially, checks can slow down the process and lead to HTTP 500 errors, so should be turned off. Defaults to TRUE.
`profile_path`	string: (optional), path to a directory to store config values in. If provided, config values will be written to a new or existing .Rprofile file for future sessions. `NULL` by default.

Value

For `galah_config()`, a `list` of all options. When `galah_config(...)` is called with arguments, nothing is returned but the configuration is set.

Examples

```
## Not run:
 galah_config()
 galah_config(caching = FALSE)
 find_reasons()
 galah_config(download_reason_id = 0,verbose = TRUE)

## End(Not run)
```

search_fields *Query layers, fields or assertions by free text search*

Description

This function can be used to find relevant fields and/or layers for use in building a set of filters with `select_filters()` or specifying required columns with `select_columns()`. This function returns a `data.frame` of all fields matching the type specified. Field names are in Darwin Core format, except in the case where the field is specific to the ALA database, in which case the ALA field name is returned.

Usage

```
search_fields(
  query,
  type = c("all", "fields", "layers", "assertions", "media", "other")
)
```

Arguments

`query`	`string`: A search string. Not case sensitive.
`type`	`string`: What type of parameters should be searched? Should be one or more of `fields`, `layers`, `assertions`, `media` or `all`.

Details

Layers are the subset of fields that are spatially appended to each record by the ALA. Layer ids are comprised of a prefix: 'el' for environmental (gridded) layers and 'cl' for contextual (polygon) layers, followed by an id number.

Value

A `data.frame` with three columns:

- id: The identifier for that layer or field. This is the value that should be used when referring to a field in another function.
- description: Detailed information on a given field
- type: Whether the field is a `field` or `layer`
- link: For layers, a link to the source data (if available)

References

- Darwin Core terms https://dwc.tdwg.org/terms/
- ALA fields https://api.ala.org.au/#ws72
- ALA assertions fields https://api.ala.org.au/#ws81

See Also

This function is used to pass valid arguments to `select_columns()` and `select_filters()`. To view valid values for a layer with categorical values, use `find_field_values()`.

Examples

```
## Not run:
test <- search_fields("species")

# Find all WorldClim layers
worldclim <- search_fields("worldclim", type = "layers")

# Return a data.frame containing all data on fields and layers
all_fields <- search_fields()

## End(Not run)
```

---

select_columns *Specify columns for occurrence download*

Description

The ALA stores content on hundreds of different fields, and users often thousands or millions of records at a time. To reduce time taken to download data, and limit complexity of the resulting `data.frame`, it is often sensible to restrict the columns returned by `ala_occurrences()` to those that are most critical for a given application. This function allows easy selection of individual columns, or commonly-requested groups of columns. The resulting `data.frame` is then passed to the `columns` argument in `ala_occurrences()`.

Usage

```
select_columns(..., group = c("basic", "event", "assertions"))
```

Arguments

`...`	zero or more individual column names to include
`group`	`string`: (optional) name of one or more column groups to include. Valid options are `"basic"`, `"event"` and `"assertion"`

Details

Calling the argument `group = "basic"` returns the following columns:

- `decimalLatitude`
- `decimalLongitude`
- `eventDate`
- `scientificName`

- taxonConceptID
- recordID
- dataResourceName

Using group = "event" returns the following columns:

- eventRemarks
- eventTime
- eventID
- eventDate
- samplingEffort
- samplingProtocol

Using group = "assertions" returns all quality assertion-related columns. The list of assertions is shown by search_fields(type = "assertions").

Value

An object of class data.frame and ala_columns specifying the name and type of each column to include in the occurrence download.

See Also

select_taxa, select_filters and select_locations for other ways to restrict the information returned by ala_occurrences and related functions.

---

select_filters *Select filters to narrow down occurrence queries*

Description

'filters' are arguments of the form field logical value that are used to narrow down the number of records returned by a specific query. For example, it is common for users to request records from a particular year (year = 2020), or to return all records except for fossils (basisOfRecord != "FossilSpecimen"). The result of select_filters can be passed to the filters argument in ala_occurrences(), ala_species() or ala_counts().

Usage

```
select_filters(..., profile = NULL)
```

Arguments

...	filters, in the form field logical value
profile	string: (optional) a data quality profile to apply to the records. See find_profiles for valid profiles. By default no profile is applied.

Details

All statements passed to `select_filters()` (except the `profile` argument) take the form of field - logical - value. Permissible examples include:

- = or == (e.g. `year = 2020`)
- `!=`, e.g. `year != 2020`)
- > or >= (e.g. `year >= 2020`)
- < or <= (e.g. `year <= 2020`)
- OR statements (e.g. `year == 2018 | year == 2020`)
- AND statements (e.g. `year >= 2000 & year <= 2020`)

In some cases R will fail to parse inputs with a single equals sign (=), particularly where statements are separated by /code& or /code|. This problem can be avoided by using a double-equals instead.

Value

An object of class `data.frame` and `ala_filters` containing filter values.

See Also

`select_taxa`, `select_columns` and `select_locations` for other ways to restrict the information returned by `ala_occurrences` and related functions. Use `search_fields` to find fields that you can filter by, and `find_field_values` to find what values of those filters are available.

Examples

```
## Not run:
# Create a custom filter for records of interest
filters <- select_filters(
    basisOfRecord = "HumanObservation",
    year >= 2010,
    stateProvince = "New South Wales")

# Add the default ALA data quality profile
filters <- select_filters(
    basisOfRecord = "HumanObservation",
    year >= 2020,
    stateProvince = "New South Wales",
    profile = "ALA")

# Use filters to exclude particular values
select_filters(year >= 2010 & year != 2021)

# Separating statements with a comma is equivalent to an 'and' statement, e.g.:
select_filters(year >= 2010 & year < 2020) # is the same as:
select_filters(year >= 2010, year < 2020)

# All statements must include the field name, e.g.
select_filters(year == 2010 | year == 2021) # this works (note double equals)
select_filters(year == 2010 | 2021) # this fails
```

```
# solr supports range queries on text as well as numbers, e.g.
select_filters(cl22 >= "Tasmania")
# queries all Australian States & Territories alphabetically after "Tasmania"

## End(Not run)
```

`select_locations` *Build a WKT string from an sf spatial object or verify an existing WKT*

Description

Restrict results to those from a specified area. Areas must be polygons and be specified as either an sf object, or a 'well-known text' (wkt) string.

Usage

```
select_locations(query)
```

Arguments

`query`	wkt string or sf object

Details

WKT strings longer than 10000 characters will not be accepted by the ALA- so the sf object or WKT string may need to be simplified.

Value

length-1 object of class `character` and `ala_locations`, containing a WKT string representing the area provided.

See Also

`select_taxa`, `select_filters` and `select_columns` for other ways to restrict the information returned by `ala_occurrences` and related functions.

Examples

```
## Not run:
# Search for records using a shapefile
locations <- select_locations(st_read(path/to/shapefile))
ala_occurrences(locations = locations)

# Search for records using a WKT
wkt <- "POLYGON((142.36228 -29.00703,142.74131 -29.00703,142.74131 \
-29.39064,142.36228 -29.39064,142.36228 -29.00703))"
ala_occurrences(locations = select_locations(wkt))

## End(Not run)
```

select_taxa *Taxon information*

Description

In the ALA, all records are associated with an identifier that uniquely identifies the taxon to which that record belongs. However, taxonomic names are often ambiguous due to homonymy; i.e. re-use of names (common or scientific) in different clades. Hence, `select_taxa` provides a means to search for taxonomic names and check the results are 'correct' before proceeded to download data via `ala_occurrences()`, `ala_species()` or `ala_counts()`. The resulting `data.frame` of taxonomic information can be passed directly to `ala_` functions to filter records to the specified taxon or taxa.

Usage

```
select_taxa(
  query,
  is_id = FALSE,
  children = FALSE,
  counts = FALSE,
  all_ranks = FALSE
)
```

Arguments

`query`	`string`: A vector containing one or more search terms, given as strings. Search terms can be scientific or common names, or taxonomic identifiers. If greater control is required to disambiguate search terms, taxonomic levels can be provided explicitly via a named `list` for a single name, or a `data.frame` for multiple names (see examples). Note that searches are not case-sensitive.
`is_id`	`logical`: Is the query a unique identifier? Defaults to `FALSE`, meaning that queries are assumed to be taxonomic names.
`children`	`logical`: Return child concepts for the provided query? DEPRECATED: use `ala_taxonomy` instead.
`counts`	`logical`: return occurrence counts for all taxa found? `FALSE` by default. DEPRECATED: use `ala_counts` instead.
`all_ranks`	`logical`: Include all available intermediate ranks for taxa? e.g. suborder, superfamily. Retrieving this information requires an additional web service call so will slow down the query. DEPRECATED: use `ala_taxonomy` instead.

Value

An object of class `data.frame` and `ala_id` containing taxonomic information.

See Also

select_columns, select_filters and select_locations for other ways to restrict the information returned by ala_occurrences and related functions. ala_taxonomy to look up taxonomic trees.

Examples

```
## Not run:
# Search using a single term
select_taxa("Reptilia")
# or equivalently:
select_taxa("reptilia") # not case sensitive

# Search using an unique taxon identifier
select_taxa(query = "https://id.biodiversity.org.au/node/apni/2914510")

# Search multiple taxa
select_taxa(c("reptilia", "mammalia")) # returns one row per taxon

## End(Not run)
```

Index