

Package ‘galah’

May 5, 2021

Type Package

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

Version 1.1.0

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'.

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

Suggests covr, dplyr, ggplot2, knitr, ozmaps, pkgdown, rmarkdown, testthat

License MPL-2.0

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

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

Maintainer Matilda Stevenson <matilda.stevenson@csiro.au>

LazyLoad yes

VignetteBuilder knitr

RoxygenNote 7.1.1

Encoding UTF-8

NeedsCompilation no

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

Repository CRAN

Date/Publication 2021-05-05 06:20:12 UTC

R topics documented:

ala_citation	2
ala_config	3
ala_counts	4
ala_media	5
ala_occurrences	6
ala_species	8
exclude	9
find_field_values	9
find_profiles	10
find_profile_attributes	11
find_reasons	11
galah	12
search_fields	14
select_columns	15
select_filters	16
select_locations	18
select_taxa	19
Index	21

ala_citation	<i>Generate a citation for occurrence data</i>
--------------	--

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:
- `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.
- `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.

Examples

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

## End(Not run)
```

ala_counts	<i>Count of ALA records</i>
------------	-----------------------------

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 or 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, filters, locations, group_by, limit = 100, type = "record")
```

Arguments

<code>taxa</code>	<code>data.frame</code> : generated by a call to <code>select_taxa()</code> . This argument also accepts a vector of unique species identifiers.
<code>filters</code>	<code>data.frame</code> : generated by a call to <code>select_filters()</code>
<code>locations</code>	string: generated by a call to <code>select_locations()</code>
<code>group_by</code>	string: field to count by
<code>limit</code>	numeric: maximum number of categories to return, defaulting to 100. If <code>limit</code> is <code>NULL</code> , all results are returned. For some categories this will take a while.
<code>type</code>	string: one of <code>c("record", "species")</code> . 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.

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, filters, locations, columns, download_dir)
```

Arguments

taxa	data.frame: generated by a call to <code>select_taxa()</code> . This argument also accepts a vector of unique species identifiers.
filters	data.frame: generated by a call to <code>select_filters()</code>
locations	string: generated by a call to <code>select_locations()</code>
columns	data.frame: generated by a call to <code>select_columns()</code>
download_dir	string: path to directory to store the downloaded media in

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 `ala_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, filters, locations, columns, mint_doi = FALSE, doi)
```

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. Specifying this argument means all other arguments will be ignored.

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 [ala_config\(\)](#).

Value

A data.frame of occurrences, columns as specified by [select_columns\(\)](#).

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, filters, locations)
```

Arguments

<code>taxa</code>	<code>data.frame</code> : generated by a call to <code>select_taxa()</code> . This argument also accepts a vector of unique species identifiers.
<code>filters</code>	<code>data.frame</code> : generated by a call to <code>select_filters()</code>
<code>locations</code>	<code>string</code> : generated by a call to <code>select_locations()</code>

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.

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)
```

exclude	<i>Negate a filter value</i>
---------	------------------------------

Description

Negate a filter 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_field_values	<i>List valid options for a categorical field</i>
-------------------	---

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("basis_of_record")
find_field_values("state")

## 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 profile 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_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 [ala_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 `ala_config()`.

galah

galah

Description

galah is an R interface to biodiversity data hosted by the Atlas of Living Australia (<https://www.ala.org.au/>) (ALA). It 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. All functions return a `data.frame` as their standard format.

Details

The Atlas of Living Australia is a repository of biodiversity data, focussed primarily on observations of individual life forms. Like the Global Biodiversity Information Facility (GBIF; <https://www.gbif.org>), the basic unit of data at ALA is an **occurrence** record, based on the 'Darwin Core' data standard (<https://dwc.tdwg.org>).

Functions that return data from ALA 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()`.

To get the most value from galah, it is helpful to understand some locally-specific terminology. 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 extension 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()`.

As the above text makes clear, 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 result of `select_taxa` to the `taxa` argument of `ala_occurrences`.

Finally, the galah package is named for the bird of the same name (*Eolophus roseicapilla*), a widely-distributed endemic Australian species.

If you have any questions, comments or suggestions, please email support@ala.org.au.

Functions

Data

- `ala_counts` Count the number of records returned by a query
- `ala_species` Download species lists
- `ala_occurrences` Download occurrence records
- `ala_media` Download images and sounds

Filter

- `select_taxa` Taxon information
- `select_filters` Filter records
- `select_locations` Specify location
- `select_columns` Columns to return in 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_reasons` List valid download reasons

Help

- [ala_config](#) Package configuration options
- [ala_citation](#) Citation for a dataset

References

For more information on the ALA API, visit <https://api.ala.org.au/>

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 = "all")
```

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	<i>Specify columns for occurrence download</i>
----------------	--

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)
```

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
- taxon_name
- taxonConceptID
- recordID
- data_resource

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

A data.frame of 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 = 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 records that are associated with a physical specimen (`basisOfRecord = "PreservedSpecimen"`). 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 = 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

By default filters are included, but they can be excluded by wrapping the filter values in [exclude](#) (see below for examples).

Value

A data.frame of 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. [exclude](#) for excluding a filter value.

Examples

```
## Not run:
# Create a custom filter for records of interest
filters <- select_filters(
  basisOfRecord = "HumanObservation",
  year = 2020,
  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
filters <- select_filters(year = exclude(seq(2011,2021)))

## End(Not run)
```

select_locations	<i>Build a WKT string from an sf spatial object or verify an existing WKT</i>
------------------	---

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

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	<i>Taxon information</i>
-------------	--------------------------

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 proceeding 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, children = FALSE, counts = 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 <code>data.frame</code> for multiple names (see examples). Note that searches are not case-sensitive.
children	logical: return child concepts for the provided query?
counts	logical: return occurrence counts for all taxa found? FALSE by default

Value

A `data.frame` of 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.

Examples

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

# Search with multiple ranks. This is required if a single term is a homonym.
select_taxa(
  list(kingdom = "Plantae", genus = "Microseris"),
  children = TRUE,
```

```
counts = TRUE)

# As above, but for multiple searches at once.
select_taxa(
  data.frame(
    genus = c("microseris", "Eucalyptus"),
    kingdom = "plantae"))

# 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

`ala_citation`, [2](#), [14](#)
`ala_config`, [3](#), [5](#), [7](#), [11](#), [12](#), [14](#)
`ala_counts`, [4](#), [6–8](#), [10](#), [12](#), [13](#), [16](#), [19](#)
`ala_media`, [5](#), [12](#), [13](#)
`ala_occurrences`, [2–5](#), [6](#), [8](#), [10–13](#), [15–19](#)
`ala_species`, [8](#), [10](#), [12](#), [13](#), [16](#), [19](#)

`exclude`, [9](#), [17](#)

`find_field_values`, [9](#), [13](#), [15](#), [17](#)
`find_profile_attributes`, [10](#), [11](#), [13](#)
`find_profiles`, [10](#), [11](#), [13](#), [17](#)
`find_reasons`, [3](#), [11](#), [13](#)

`galah`, [12](#)

`search_fields`, [9](#), [10](#), [12](#), [13](#), [14](#), [17](#)
`select_columns`, [5](#), [7](#), [13–15](#), [15](#), [17–19](#)
`select_filters`, [4](#), [5](#), [7–11](#), [13–16](#), [16](#), [18](#), [19](#)
`select_locations`, [4](#), [5](#), [7](#), [8](#), [13](#), [16](#), [17](#), [18](#),
[19](#)
`select_taxa`, [4](#), [5](#), [7–9](#), [13](#), [16–18](#), [19](#)