Package 'ALA4R'

February 18, 2017

Type Package

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

Version 1.5.6

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 <http://ala.org.au/>. ALA4R 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. Our goal is to enable outputs (e.g. observations of species) to be queried and output in a range of standard formats.

Imports RCurl, assertthat, digest, httr, jsonlite (>= 0.9.8), plyr, sp, stringr (>= 1.0.0), wellknown

Suggests ape, covr, data.table, geosphere, jpeg, knitr, mapdata, maps, maptools, mgcv, phytools, rmarkdown, testthat, vegan

License MPL-2.0

URL https://github.com/AtlasOfLivingAustralia/ALA4R

LazyLoad yes

LazyData yes

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Ben Raymond [aut, cre],
Jeremy VanDerWal [aut],
Lee Belbin [aut],
Michael Sumner [ctb],
Tom August [ctb],
John Baumgartner [ctb]

Maintainer Ben Raymond <ben_ala@untan.gl>

Repository CRAN

Date/Publication 2017-02-18 09:38:25

R topics documented:

ALA4R **ALA4R**

Description

This project enables the R community to access data and tools hosted by the Atlas of Living Australia. The goal of the project is to enable basic species and related information to be queried and output in standard formats for R. ALA4R is based around the extensive web services provided by the Atlas; see the API link below.

References

http://api.ala.org.au/

`ala_cache_filename` *Returns the name of the cache file associated with the given URL. Note that this file may not actually exist, this function just provides the mapping from URL to filename*

Description

Returns the name of the cache file associated with the given URL. Note that this file may not actually exist, this function just provides the mapping from URL to filename

Usage

```
ala_cache_filename(url)
```

Arguments

`url`	string: the URL

Value

string: the file path and name

References

http://api.ala.org.au/

See Also

`ala_config` for cache settings, particularly the cache directory

Examples

```
ala_cache_filename("http://biocache.ala.org.au/ws/index/fields")
```

`ala_config` *Get or set configuration options that control ALA4R behaviour*

Description

Get or set configuration options that control ALA4R behaviour

Usage

```
ala_config(...)

ala_reasons()
```

Arguments

...	Options can be defined using name=value. Valid options are: • reset: `ala_config("reset")` will reset the options to their default values • caching string: caching can be "on" (results will be cached, and any cached results will be re-used), "refresh" (cached results will be refreshed and the new results stored in the cache), or "off" (no caching, default). • 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. • verbose logical: should ALA4R give verbose output to assist debugging? (default=FALSE) • warn_on_empty logical: should a warning be issued if a request returns an empty result set? (default=FALSE) • user_agent string: the user-agent string used with all web requests to the ALA servers. Default = "ALA4R" with version number • text_encoding string: text encoding assumed when reading cached files from local disk (default="UTF-8") • 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 `ala_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.

Value

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

References

http://api.ala.org.au/

http://spatial.ala.org.au/layers-service/ this will eventually move to the api link

Invoking `ala_config()` with no arguments returns a list with the current values of the options.

`ala_reasons()` returns a data frame with information describing the valid options for `download_reason_id`

Examples

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

## End(Not run)
```

ala_fields *Retrieves a list of all field names that can be used with data retrieval functions*

Description

Note for occurrence fields: only fields that are indexed in the ALA database can be queried (e.g. used in the `fq` parameter in `occurrences`. These fields are identified by the `indexed` column in `ala_fields("occurrence")`. Only fields that are stored in the database can be returned as part of an `occurrences` call. These fields are identified by the `stored` column in `ala_fields("occurrence")`. The calling syntaxes `ala_fields("occurrence_stored")` and `ala_fields("occurrence_indexed")` are for convenience, and are equivalent to `subset(ala_fields("occurrence"),stored)` and `subset(ala_fields("occurrence"),indexed)`.

Usage

```
ala_fields(fields_type = "occurrence", as_is = TRUE)

field_info(field_id, maxrows = 50, record_count_only = FALSE)
```

Arguments

`fields_type`	text: one of the following • "general" - for searching taxa, datasets, layers, and collections metadata • "occurrence" - for species occurrence records • "occurrence_stored" - can be returned as part of a species occurrence record search (equivalant to `subset(ala_fields("occurrences"),stored)`) • "occurrence_indexed" - can be queried as part of a species occurrence record search (equivalant to `subset(ala_fields("occurrences"),indexed)`) • "layers" - fields associated with the environmental and contextual layers. For additional information on layers, including metadata and licensing, see `search_layers` • "assertions" - potential issues flagged on one or more occurrence record fields
`as_is`	logical: if TRUE, leave the field names as they are returned from the ALA web services. Arguments that are passed directly to the ALA's web services (e.g. parameter `fq` in `occurrences`) should use field names in this format. If `as_is` is FALSE, the returned $names entries will be modified to make them consistent with the corresponding column names in R data.frames returned by e.g. `occurrences`. `as_is=FALSE` has no effect when `fields_type` is "layers". Note that prior to v1.20, `as_is=FALSE` did not work correctly.
`field_id`	text: id of environmental/contextual layer field for which to look up information Prepend "el" for "environmental" (gridded) layers and "cl" for "contextual" (polygonal) layers

`maxrows` integer: maximum number of records to download. Some contextual layers (those with `field_ids` starting with "cl") have a very large number of records and attempting to download the full set can cause R to crash. Specifying -1 for maxrows will download the full set of records for that field

`record_count_only` logical: if TRUE, return just the count of records that would be downloaded, but don't download them. This really only makes sense for contextual layers, because environmental layers have only one record per layer

Value

If `record_count_only` is TRUE, the number of records is returned as numeric. Otherwise, a data frame containing the field name and various attributes; an empty data frame is returned if no match is found

References

Relevant ALA web services:

- for fields_type "occurrence": http://api.ala.org.au/#ws72
- for fields_type "general": http://api.ala.org.au/#ws88
- for fields_type "layers": http://api.ala.org.au/#ws11 (see also descriptions of the spatial layers: http://spatial.ala.org.au/layers/)
- for fields_type "assertions": http://api.ala.org.au/#ws81

See Also

`search_layers` to search for spatial layers

Examples

```
## Not run:
 l <- ala_fields("layers")
 l[,4]
 o <- ala_fields("occurrence")
 o[1:13,]
 a <- ala_fields("assertions")
 a$description
 field_info("cl22")
 field_info("el773")

## End(Not run)
```

ala_list *Species lists*

Description

Note that this refers to pre-generated lists of species stored on the ALA servers. The similarly-named but different function specieslist provides a different function, namely listing the species matching a query or recorded as present in a search area.

Usage

```
ala_list(druid, kvp = TRUE, verbose = ala_config()$verbose)
```

Arguments

druid	string: data resource UID of the list (i.e. the list identifier)
kvp	logical: include key-value pairs? Some lists contain information about the species in the form of key-value pairs
verbose	logical: show additional progress information?

Value

data.frame

References

http://lists.ala.org.au and the associated web services at http://lists.ala.org.au/ws

See Also

specieslist ala_lists

Examples

```
## Not run:
 all_lists <- ala_lists()
 ## find the "Field Guide apps species profiles" from Museum Victoria
 all_lists[grep("Field Guide",all_lists$listName),]
 ## download the vertebrates one
 l <- ala_list(druid="dr1146")

## End(Not run)
```

`ala_lists` *Find ALA species lists*

Description

Find ALA species lists

Usage

```
ala_lists(guid, offset = 0, max = 500, verbose = ala_config()$verbose)
```

Arguments

`guid`	string: (optional) if provided, return only lists in which this GUID appears
`offset`	integer: the number of lists to skip. This supports paging
`max`	integer: the maximum number of lists to return. This supports paging
`verbose`	logical: show additional progress information?

Value

data.frame of list name and other details

References

http://lists.ala.org.au and the associated web services at http://lists.ala.org.au/ws

See Also

ala_list

Examples

```
## Not run:
 ## lists that include the giant African snail Achatina fulica
 ##  (which is a notifiable pest species in some states)
 l <- ala_lists(search_guids("Achatina fulica")$guid)

## End(Not run)
```

aus

Basic Australia raster for visualization A basic raster dataset in SpatialGridDataFrame (package sp) representing Australia at 0.05 degree (~5km) resolution.

Description

Basic Australia raster for visualization A basic raster dataset in SpatialGridDataFrame (package sp) representing Australia at 0.05 degree (~5km) resolution.

Usage

```
data(aus)
```

Format

SpatialGridDataFrame (package sp)

check_assertions

Check assertions in occurrences object

Description

This provides a data.frame detailing the assertions that are found in a dataset returned from occurrences.

Usage

```
check_assertions(x)
```

Arguments

x	list: an object returned from occurrences

Value

A dataframe of assertions column names, descriptions and categories/error codes. If no assertions are in the dataset, NULL is returned.

References

http://api.ala.org.au/, 'http://biocache.ala.org.au/ws/assertions/codes'

Examples

```
#download species data with all possible assertions
## Not run:
 x <- occurrences(taxon="golden bowerbird",download_reason_id=10,qa=ala_fields("assertions")$name)
 asserts <- check_assertions(x) #data.frame of assertions, their description and column names
 asserts$description # List out descriptions of all (current) assertions

 tmp <- x$data[,names(x$data) %in% asserts$name] ## assertion columns from data
 which(colSums(tmp)>0) ## discard those not seen in the data

## End(Not run)
```

`fieldguide` *Generate a PDF field guide using the ALA's field guide generator*

Description

Generate a PDF field guide using the ALA's field guide generator

Usage

```
fieldguide(guids, title = "Field guide", filename = tempfile(fileext =
  ".pdf"), overwrite = FALSE)
```

Arguments

`guids`	character: vector of GUIDs
`title`	string: title to use in the field guide PDF
`filename`	string: filename for the PDF document
`overwrite`	logical: overwrite the file if it already exists?

Value

filename

References

http://fieldguide.ala.org.au/

See Also

search_guids

Examples

```
## Not run:
fieldguide(guids=
  c("urn:lsid:biodiversity.org.au:afd.taxon:95773568-053d-44de-a624-5699f0ac4a59",
  "http://id.biodiversity.org.au/node/apni/2890970"))

## End(Not run)
```

`image_info` *Fetch information about an image, given its image ID*

Description

Note that there is currently no web service that provides image information, and so we are scraping results from pages of the form http://images.ala.org.au/image/details?imageId=id. This web scraping may be fragile, and will be replaced by a web-service-based function when one becomes available.

Usage

```
image_info(id, image_number, verbose = ala_config()$verbose)
```

Arguments

`id`	character: IDs of images (e.g. as returned by `occurrences` in the imageUrl column). Each ID will be of a format something like "84654e14-dc35-4486-9e7c-40eb2f8d3faa"
`image_number`	character or numeric: ID numbers of images (e.g. as returned by ALA's image search at `http://images.ala.org.au/`. Each image_number will be of a format something like 122218480
`verbose`	logical: show additional progress information? [default is set by ala_config()]

Value

A data.frame with one row per `id`, and at least the columns imageIdentifier and imageURL

See Also

`ala_config` `occurrences`

Examples

```
## Not run:
image_info(c("84654e14-dc35-4486-9e7c-40eb2f8d3faa",
  "39836d30-0761-473d-bac2-9ed9494fd37e",
  "this-is-an-invalid-image-id"))

## End(Not run)
```

`intersect_points` *Intersect environmental or contextual layers at a given a set of points (coordinates)*

Description

Intersect environmental or contextual layers at a given a set of points (coordinates)

Usage

```
intersect_points(pnts, layers, SPdata.frame = FALSE, use_layer_names = TRUE,
  verbose = ala_config()$verbose)
```

Arguments

`pnts`	numeric: vector of latitude/longitude pairs, or a 2 column data.frame or matrix of lat,lons. NOTE: the number of locations must be less than 100000.
`layers`	string vector: ids of layers to be intersected. The list of possible layers is available from `ala_fields("layers")`. Names can be passed as full layer names (e.g. "Radiation - lowest period (Bio22)") rather than id ("el871"). Note: if more than one location has been provided in `pnts`, the number of layers must be less than 700.
`SPdata.frame`	logical: should the output should be returned as a SpatialPointsDataFrame of the sp package or simply as a data.frame?
`use_layer_names`	logical: if TRUE, layer names will be used as column names in the returned data frame (e.g. "radiationLowestPeriodBio22"). Otherwise, layer id value will be used for column names (e.g. "el871")
`verbose`	logical: show additional progress information? [default is set by `ala_config`]

Value

A SpatialPointsDataFrame containing the intersecting data information. Missing data or incorrectly identified layer id values will result in NA data

References

The associated ALA web service: http://api.ala.org.au/#ws84

Descriptions of the spatial layers: http://spatial.ala.org.au/layers/

This function allows the user to sample environmental/contextual layers at arbitrary locations. It complements the `occurrences` function, which allows values of the same set of layers to be downloaded at species occurrence locations. NOTE: batch requests (multiple points) are currently processed in a *single queue* on the ALA servers. Processing times may be slow if there are many requests in the queue. Note also that the actual processing of batch requests is inherently slow: a large number of points may take quite some time. Be warned.

See Also

ala_config

Examples

```
## Not run:
 #single point with multiple layers
 layers <- c('cl22','cl23','el773')
 pnts <- c(-23.1,149.1)
 intersect_points(pnts,layers)
 # equivalent direct web service call:
 # http://spatial.ala.org.au/ws/intersect/cl22,cl23,el773/-23.1/149.1

 ## multiple points as a grid sampling multiple layers
 layers <- c('cl22','cl23','el773')
 pnts <- data.frame(expand.grid(lat=seq(-29,-19,2.0),lon=seq(130.0,140.0,2.0)))
 intersect_points(pnts,layers)

## End(Not run)
```

---

occurrences *Get occurrence data*

Description

Retrieve ALA occurrence data via the "occurrence download" web service. At least one of `taxon`, `wkt`, or `fq` must be supplied for a valid query. Note that there is a limit of 500000 records per request when using `method="indexed"`. Use the `method="offline"` for larger requests. For small requests, `method="indexed"` may be faster.

Usage

```
occurrences(taxon, wkt, fq, fields, extra, qa, method = "indexed", email,
  download_reason_id = ala_config()$download_reason_id, reason,
  verbose = ala_config()$verbose, record_count_only = FALSE,
  use_layer_names = TRUE, use_data_table = TRUE)
```

Arguments

`taxon`	string: (optional) query of the form field:value (e.g. "genus:Macropus") or a free text search ("Alaba vibex")
`wkt`	string: (optional) a WKT (well-known text) string providing a spatial polygon within which to search, e.g. "POLYGON((140 -37,151 -37,151 -26,140.131 -26,140 -37))"
`fq`	string: (optional) character string or vector of strings, specifying filters to be applied to the original query. These are of the form "INDEXEDFIELD:VALUE" e.g. "kingdom:Fungi". See `ala_fields("occurrence_indexed",as_is=TRUE)`

for all the fields that are queryable. NOTE that fq matches are case-sensitive, but sometimes the entries in the fields are not consistent in terms of case (e.g. kingdom names "Fungi" and "Plantae" but "ANIMALIA"). fq matches are ANDed by default (e.g. c("field1:abc","field2:def") will match records that have field1 value "abc" and field2 value "def"). To obtain OR behaviour, use the form c("field1:abc OR field2:def"). See e.g. http://wiki.apache.org/solr/CommonQueryParameters for more information about filter queries

`fields` string vector: (optional) a vector of field names to return. Note that the columns of the returned data frame are not guaranteed to retain the ordering of the field names given here. If not specified, a default list of fields will be returned. See `ala_fields("occurrence_stored")` for valid field names with method `indexed`, and `ala_fields("occurrence")` for valid field names with method `offline`. Field names can be passed as full names (e.g. "Radiation - lowest period (Bio22)") rather than id ("el871"). Use `fields="all"` to include all available fields, but note that `"all"` will probably cause an error with `method="offline"` because the request URL will exceed the maximum allowable length

`extra` string vector: (optional) a vector of field names to include in addition to those specified in `fields`. This is useful if you would like the default list of fields (i.e. when `fields` parameter is not specified) plus some additional extras. See `ala_fields("occurrence_stored",as_is=TRUE)` for valid field names. Field names can be passed as full names (e.g. "Radiation - lowest period (Bio22)") rather than id ("el871"). Use `extra="all"` to include all available fields, but note that `"all"` will probably cause an error with `method="offline"` because the request URL will exceed the maximum allowable length

`qa` string vector: (optional) list of record issues to include in the download. Use `qa="all"` to include all available issues, or `qa="none"` to include none. Otherwise see `ala_fields("assertions",as_is=TRUE)` for valid values

`method` string: "indexed" (default) or "offline". In "offline" mode, more fields are available and larger datasets can be returned

`email` string: the email address of the user performing the download (required for `method="offline"`

`download_reason_id`
numeric or string: (required unless record_count_only is TRUE) a reason code for the download, either as a numeric ID (currently 0–11) or a string (see `ala_reasons` for a list of valid ID codes and names). The download_reason_id can be passed directly to this function, or alternatively set using `ala_config(download_reason_id=...)`

`reason` string: (optional) user-supplied description of the reason for the download. Providing this information is optional but will help the ALA to better support users by building a better understanding of user communities and their data requests

`verbose` logical: show additional progress information? [default is set by ala_config()]

`record_count_only`
logical: if TRUE, return just the count of records that would be downloaded, but don't download them. Note that the record count is always re-retrieved from the ALA, regardless of the caching settings. If a cached copy of this query exists on the local machine, the actual data set size may therefore differ from this record count. `record_count_only=TRUE` can only be used with `method="indexed"`

use_layer_names
: logical: if TRUE, layer names will be used as layer column names in the returned data frame (e.g. "radiationLowestPeriodBio22"). Otherwise, layer id value will be used for layer column names (e.g. "el871")

use_data_table
: logical: if TRUE, attempt to read the data.csv file using the fread function from the data.table package. Requires data.table to be available. If this fails with an error or warning, or if use_data_table is FALSE, then read.table will be used (which may be slower)

Value

Data frame of occurrence results, with one row per occurrence record. The columns of the dataframe will depend on the requested fields

References

- Associated ALA web service for record counts: http://api.ala.org.au/#ws3
- Associated ALA web service for occurence downloads: http://api.ala.org.au/#ws4
- Field definitions: https://docs.google.com/spreadsheet/ccc?key=0AjNtzhUIIHeNdHhtcFVSM09qZ3c3N3ItUnBE
- WKT reference: http://www.geoapi.org/3.0/javadoc/org/opengis/referencing/doc-files/WKT.html

See Also

ala_reasons for download reasons; ala_config

Examples

```
## Not run:
## count of records from this data provider
x <- occurrences(taxon="data_resource_uid:dr356",record_count_only=TRUE)
## download records, with standard fields
x <- occurrences(taxon="data_resource_uid:dr356",download_reason_id=10)
## download records, with all fields
x <- occurrences(taxon="data_resource_uid:dr356",download_reason_id=10,
  fields=ala_fields("occurrence_stored",as_is=TRUE)$name)
## download records, with specified fields
x <- occurrences(taxon="macropus",fields=c("longitude","latitude","common_name",
  "taxon_name","el807"),download_reason_id=10)
 ## download records in polygon, with no quality assertion information
x <- occurrences(taxon="macropus",
  wkt="POLYGON((145 -37,150 -37,150 -30,145 -30,145 -37))",
  download_reason_id=10,qa="none")

y <- occurrences(taxon="alaba vibex",fields=c("latitude","longitude","el874"),download_reason_id=10)
str(y)
# equivalent direct webservice call:
# http://biocache.ala.org.au/ws/occurrences/index/download?reasonTypeId=10&q=Alaba%20vibex&
#    fields=latitude,longitude,el874&qa=none
```

```
occurrences(taxon="Eucalyptus gunnii",fields=c("latitude","longitude"),
  qa="none",fq="basis_of_record:LivingSpecimen",download_reason_id=10)
# equivalent direct webservice call:
# http://biocache.ala.org.au/ws/occurrences/index/download?reasonTypeId=10&q=Eucalyptus%20gunnii&
#    fields=latitude,longitude&qa=none&fq=basis_of_record:LivingSpecimen

## End(Not run)
```

`occurrences_plot` *Quick geographic plot of occurrence data*

Description

Generates a plot of occurrence data retrieved using occurrences.The plot uses an Australian basemap and colours the occurrence records dots according to parameters

Usage

```
occurrences_plot(x, filename = "Rplots.pdf", qa = c("fatal", "error"),
  grouped = FALSE, taxon_level = "species", pch, cex = 0.75, ...)
```

Arguments

x	list: a list object that has been downloaded using occurrences
filename	string: name of file to be created; defaults to RPlots.pdf
qa	string vector: list of record issues to be mapped; these can be assertion column-names, or 'all' or 'none' or any combination of 'error', 'warning' or 'fatal'. Column or categories in your dataset can be viewed using check_assertions.
grouped	logical: TRUE creates a single plot for all observations; FALSE plots individual maps for the taxon level defined.
taxon_level	string: taxonomic level at which to create maps; possible values are 'species', 'genus', 'family' or 'order'
pch	single number or character representing point type. See description of pch in points.
cex	numeric: character (or symbol) expansion. See description of cex in points.
...	: other options passed to pdf()

Value

Generates a pdf that maps the distributions.

References

http://api.ala.org.au/

Examples

```
## Not run:
#download some observations
x <- occurrences(taxon="Eucalyptus gunnii",download_reason_id=10)
occurrences_plot(x)
x <- occurrences(taxon="Cider Gum",download_reason_id=10)
occurrences_plot(x,"alaPlot.pdf",qa="fatal",grouped=FALSE, taxon_level="species",pch='+')

## End(Not run)
```

`occurrences_s3` *Summarize, filter and subset occurrence data*

Description

Set of S3 methods to summarize, filter and get unique occurrence data retrieved using `occurrences`. This uses information based on selections of assertions (quality assurance issues ALA has identified), spatial and temporal data.

Usage

```
## S3 method for class 'occurrences'
summary(object, ...)

## S3 method for class 'occurrences'
unique(x, incomparables = FALSE, spatial = 0,
  temporal = NULL, na.rm = FALSE, ...)

## S3 method for class 'occurrences'
subset(x, remove.fatal = TRUE,
  exclude.spatial = "error", exclude.temporal = "error",
  exclude.taxonomic = "error", max.spatial.uncertainty,
  keep.missing.spatial.uncertainty = TRUE, ...)
```

Arguments

`object`	list: an 'occurrence' object that has been downloaded using `occurrences`
`...`	not currently used
`x`	list: an 'occurrence' object that has been downloaded using `occurrences`
`incomparables`	logical/numeric: currently ignored, but needed for S3 method consistency
`spatial`	numeric: specifies a rounding value in decimal degrees used to to create a unique subset of the data. Value of 0 means no rounding and use values as is. Values <0 mean ignore spatial unique parameter
`temporal`	character: specifies the temporal unit for which to keep unique records; this can be by "year", "month", "yearmonth" or "full" date. NULL means ignore temporal unique parameter

`na.rm` logical: keep (FALSE) or remove (TRUE) missing spatial or temporal data

`remove.fatal` logical: remove flagged assertion issues that are considered "fatal"; see `check_assertions`

`exclude.spatial` character vector: defining flagged spatial assertion issues to be removed. Values can include 'warnings','error','missing','none'; see `check_assertions`

`exclude.temporal` character vector: defining flagged temporal assertion issues to be removed. Values can include 'warnings','error','missing','none'; see `check_assertions`

`exclude.taxonomic` character vector: defining flagged taxonomic assertion issues to be removed. Values can include 'warnings','error','missing','none'; see `check_assertions`

`max.spatial.uncertainty` numeric: number defining the maximum spatial uncertainty (in meters) one is willing to accept.

`keep.missing.spatial.uncertainty` logical: keep (FALSE) or remove (TRUE) information missing spatial uncertainty data.

Details

`unique` will give the min value for all columns that are not used in the aggregation.

References

http://api.ala.org.au/

http://stat.ethz.ch/R-manual/R-devel/library/methods/html/Methods.html

Examples

```
## Not run:
#download some observations
x <- occurrences(taxon="Amblyornis newtonianus",download_reason_id=10)

#summarize the occurrences
summary(x)

#keep spatially unique data at 0.01 degrees (latitude and longitude)
tt <- unique(x,spatial=0.01)
summary(tt)

#keep spatially unique data that is also unique year/month for the collection date
tt <- unique(x,spatial=0,temporal='yearmonth')
summary(tt)

#keep only information for which fatal or "error" assertions do not exist
tt <- subset(x)
summary(tt)

## End(Not run)
```

occurrence_details *Retrieve the full details of occurrence records*

Description

Note that this makes a separate web request for each occurrence uuid, and so may not be wise to use on a large number of uuids.

Usage

```
occurrence_details(uuid, verbose = ala_config()$verbose)
```

Arguments

uuid	string: one or more record ids, as returned by occurrences (in the data$id column)
verbose	logical: show additional progress information? [default is set by ala_config()]

Value

A named list (named by uuid), each element of which is a list containing the details for that uuid. This inner list will be empty if no match is found for the supplied uuid

References

Associated ALA web service: http://api.ala.org.au/#ws102

See Also

occurrences ala_config

Examples

```
## Not run:
s1 <- occurrence_details("f259c5ce-200c-41a2-b73a-e36a91f748f7")
str(s1,max.level=3)

## End(Not run)
```

search_fulltext *Full text search*

Description

Performs a search across all objects, and selects the closest matches. Generally, the user will provide the search term via the query parameter, with optional filtering via fq.

Usage

```
search_fulltext(query, fq, output_format = "simple", start, page_size,
  sort_by, sort_dir)
```

Arguments

query	string: the search term
fq	string: (optional) character string or vector of strings, specifying filters to be applied to the original query. These are of the form "INDEXEDFIELD:VALUE" e.g. "rk_kingdom:Fungi". See ala_fields("general") for all the fields that are queryable. NOTE that fq matches are case-sensitive, but sometimes the entries in the fields are not consistent in terms of case (e.g. kingdom names "Fungi" and "Plantae" but "ANIMALIA"). fq matches are ANDed by default (e.g. c("field1:abc","field2:def") will match records that have field1 value "abc" and field2 value "def"). To obtain OR behaviour, use the form c("field1:abc OR field2:def")
output_format	string: controls the print method for the "data" component of the returned object. Either "complete" (the complete data structure is displayed), or "simple" (a simplified version is displayed). Note that the complete data structure exists in both cases: this option only controls what is displayed when the object is printed to the console. The default output format is "simple"
start	numeric: (optional) (positive integer) start offset for the results
page_size	numeric: (optional) (positive integer) maximum number of records to return. Defaults to the server-side value - currently 10
sort_by	string: (optional) field to sort on. See ala_fields("general") for valid field names
sort_dir	string: (optional) sort direction, either "asc" or "desc"

Value

a named list with the components "meta" (search metadata), "facets" (search facet results), and "data" (the search results, in the form of a data frame). The contents (column names) of the data frame will vary depending on the details of the search and the results, but should contain at least the columns guid, name, commonName, rank, author, and occurrenceCount.

References

Associated ALA web service: http://api.ala.org.au/#ws1

See Also

ala_fields

Examples

```
## Not run:
 # find information ALA holds on red kangaroo
 search_fulltext("red kangaroo")
 search_fulltext("Macropus rufus")
 search_fulltext("urn:lsid:biodiversity.org.au:afd.taxon:31a9b8b8-4e8f-4343-a15f-2ed24e0bf1ae")

 # find genus names like "Oenanthe"
 search_fulltext("oenanthe",sort_by="rk_kingdom",fq="rank:genus")

## End(Not run)
```

---

search_guids *Lookup of species identifiers*

Description

Provides names, taxonomic classification, and other information for a list of GUIDs.

Usage

```
search_guids(guids = c(), occurrence_count = FALSE,
  output_format = "simple")
```

Arguments

guids	string: a single GUID or vector of GUIDs
occurrence_count	logical: if TRUE then also return the number of occurrences of each matched GUID. Note that this requires one extra web call for each GUID, and so may be slow.
output_format	string: controls the print method for the returned object. Either "complete" (the complete data structure is displayed), or "simple" (a simplified version is displayed). Note that the complete data structure exists in both cases: this option only controls what is displayed when the object is printed to the console. The default output format is "simple"

Value

A data frame, which should include one entry (i.e. one data.frame row or one list element) per input GUID. The columns in the data.frame output may vary depending on the results returned by the ALA server, but should include searchTerm, name, rank, and guid.

References

The associated ALA web service: http://api.ala.org.au/#ws87

Examples

```
## Not run:
search_guids(c("urn:lsid:biodiversity.org.au:afd.taxon:95773568-053d-44de-a624-5699f0ac4a59",
  "http://id.biodiversity.org.au/node/apni/2890970","this_is_not_a_valid_guid"))

## End(Not run)
```

---

search_layers *Search for environmental and contextual data layers*

Description

Search for environmental and contextual data layers

Usage

```
search_layers(query, type = "all", output_format = "simple")
```

Arguments

query	text string: optional search term against layer metadata. Only layers that include this term in their metadata will be returned.
type	string: either "all" (all possible layers; default), "grids" (gridded environmental layers), or "shapes" (contextual shapefile layers)
output_format	string: controls the print method for the returned object. Either "complete" (the complete data structure is displayed), or "simple" (a simplified version is displayed). Note that the complete data structure exists in both cases: this option only controls what is displayed when the object is printed to the console. The default output format is "simple"

Value

A data frame of results. The contents (column names) of the data frame will vary depending on the details of the search and the results

References

Associated ALA web services: http://api.ala.org.au/#ws11 http://api.ala.org.au/#ws12 http://api.ala.org.au/#ws13

Descriptions of the spatial layers: http://spatial.ala.org.au/layers/)

Examples

```
## Not run:
search_layers(type="all")
search_layers(type="grids",query="income")
search_layers(type="shapes",query="coral",output_format="simple")

## End(Not run)
```

---

search_names *Lookup of taxonomic names*

Description

Provides GUID, taxonomic classification, and other information for a list of names. Case-insensitive but otherwise exact matches are used.

Usage

```
search_names(taxa = c(), vernacular = FALSE, guids_only = FALSE,
  occurrence_count = FALSE, output_format = "simple")
```

Arguments

taxa	string: a single name or vector of names
vernacular	logical: if TRUE, match on common names as well as scientific names, otherwise match only on scientific names
guids_only	logical: if TRUE, a named list of GUIDs will be returned. Otherwise, a data frame with more comprehensive information for each name will be returned.
occurrence_count	logical: if TRUE (and if guids_only is FALSE) then also return the number of occurrences of each matched name. Note that this requires one extra web call for each name, and so may be slow. Only applicable if guids_only is FALSE.
output_format	string: controls the print method for the returned object (only has an effect when guids_only is FALSE). Either "complete" (the complete data structure is displayed), or "simple" (a simplified version is displayed). Note that the complete data structure exists in both cases: this option only controls what is displayed when the object is printed to the console. The default output format is "simple"

Value

A data frame of results, or named list of GUIDs if `guids_only` is TRUE. The results should include one entry (i.e. one data.frame row or one list element) per input name. The columns in the data.frame output may vary depending on the results returned by the ALA server, but should include searchTerm, name, rank, and guid.

References

The associated ALA web service: http://api.ala.org.au/#ws87

Examples

```
## Not run:
search_names(c("Grevillea humilis","Grevillea humilis subsp. maritima",
  "Macropus","Thisisnot aname"))

search_names(c("Grevillea humilis","Grevillea humilis subsp. maritima",
  "Macropus","Thisisnot aname"),guids_only=TRUE)

search_names("kookaburra",vernacular=FALSE)

search_names("kookaburra",vernacular=TRUE)

## occurrence counts for matched names
search_names(c("Grevillea humilis","Grevillea humilis subsp. maritima",
  "Macropus","Thisisnot aname"),occurrence_count=TRUE)

## no occurrence counts because guids_only is TRUE
search_names(c("Grevillea humilis","Grevillea humilis subsp. maritima",
  "Macropus","Thisisnot aname"),occurrence_count=TRUE,guids_only=TRUE)

## End(Not run)
```

---

search_partial_name *Partial-name search*

Description

A partial-name search for species names & identifiers used at the ALA. If searching for a taxon name, and the scientific name or common name of the taxon are known, use `search_names` instead.

Usage

```
search_partial_name(taxon, geo_only = FALSE, output_format = "simple",
  index_type, limit)
```

Arguments

taxon	string: part of the scientific, common name of the taxa
geo_only	logical: if TRUE, only results that have geospatial occurrence records will be included
output_format	string: controls the print method for the returned object. Either "complete" (the complete data structure is displayed), or "simple" (a simplified version is displayed). Note that the complete data structure exists in both cases: this option only controls what is displayed when the object is printed to the console. The default output format is "simple"
index_type	string: the index type to limit. Values include: TAXON REGION COLLECTION INSTITUTION DATASET. By default, no index_type limit is applied
limit	numeric: the maximum number of matches returned (defaults to the server-side value - currently 10)

Value

A dataframe of results. The contents (column names) of the data frame will vary depending on the details of the search and the results.

References

Associated ALA web service: http://api.ala.org.au/#ws25

See Also

search_names for searching known scientific or common taxonomic names

Examples

```
## Not run:
# find any names containing "allaba"
search_partial_name("allaba",output_format="simple")

# retrieve only species that have geolocated occurrence records
search_partial_name("Gallaba",geo_only=TRUE)

## End(Not run)
```

---

sites_by_species *Sites by species*

Description

A data.frame is returned as grid cells by species with values in each cell being the number of occurrences of each species. No null (all zero) species should be returned. The coordinates returned are the TOP-LEFT corner of the grid cell.

Usage

```
sites_by_species(taxon, wkt, gridsize = 0.1, SPdata.frame = FALSE,
  verbose = ala_config()$verbose)
```

Arguments

taxon	string: the identifier to get the species data from the ala biocache. E.g. "genus:Macropus". See `ala_fields("occurrence_stored")` for valid field names
wkt	string: Bounding area in Well Known Text (WKT) format. E.g. "POLYGON((118 -30,146 -30,146 -11,118 -11,118 -30))".
gridsize	numeric: size of output grid cells in decimal degrees. E.g. 0.1 (=~10km)
SPdata.frame	logical: should the output be returned as a SpatialPointsDataFrame of the sp package?
verbose	logical: show additional progress information? [default is set by ala_config()]

Value

A dataframe or a SpatialPointsDataFrame containing the species by sites data. Columns will include longitude, latitude, and each species present. Values for species are record counts (i.e. number of recorded occurrences of that taxon in each grid cell). The `guid` attribute of the data frame gives the guids of the species (columns) as a named character vector

References

Associated web services: http://spatial.ala.org.au/ws

http://www.geoapi.org/3.0/javadoc/org/opengis/referencing/doc-files/WKT.html

Examples

```
## Not run:
# Eucalyptus in Tasmania based on a 0.1 degree grid
ss <- sites_by_species(taxon="genus:Eucalyptus",wkt="POLYGON((144 -43,148 -43,148 -40,
  144 -40,144 -43))",gridsize=0.1,verbose=TRUE)
head(ss[,1:6])
# equivalent direct POST webservice call:
# http://spatial.ala.org.au/alaspatial/ws/sitesbyspecies?speciesq=genus:Eucalyptus&qname=data&
#  area=POLYGON((144%20-43,148%20-43,148%20-40,144%20-40,144%20-43))&bs=http://biocache.ala.org.au/
#  ws/&movingaveragesize=1&gridsize=0.1&sitesbyspecies=1

## get the guid of the first species (which is the third column of the data frame, since the
##  first two columns are longitude and latitude)
attr(ss,"guid")[1]

## End(Not run)
```

specieslist *Get list of taxa and their occurrence counts*

Description

Retrieve a list of taxa matching a search query, within a spatial search area, or both

Usage

```
specieslist(taxon, wkt, fq)
```

Arguments

taxon	string: Text of taxon, e.g. "Macropus rufus" or "macropodidae"
wkt	string: WKT (well-known text) defining a polygon within which to limit taxon search, e.g. "POLYGON((140 -37,151 -37,151 -26,140 -26,140 -37))"
fq	string: character string or vector of strings, specifying filters to be applied to the original query. These are of the form "INDEXEDFIELD:VALUE" e.g. "kingdom:Fungi". See `ala_fields("occurrence_indexed",as_is=TRUE)` for all the fields that are queryable. NOTE that fq matches are case-sensitive, but sometimes the entries in the fields are not consistent in terms of case (e.g. kingdom names "Fungi" and "Plantae" but "ANIMALIA"). fq matches are ANDed by default (e.g. c("field1:abc","field2:def") will match records that have field1 value "abc" and field2 value "def"). To obtain OR behaviour, use the form c("field1:abc OR field2:def")

Value

data frame of results, where each row is a taxon, its classification information, and its occurrence count

References

Associated ALA web service: http://api.ala.org.au/#ws98

http://www.geoapi.org/3.0/javadoc/org/opengis/referencing/doc-files/WKT.html

See Also

ala_fields for occurrence fields that are queryable via the fq parameter

Examples

```
## Not run:
x <- specieslist(taxon="macropus",wkt="POLYGON((145 -37,150 -37,150 -30,145 -30,145 -37))")

x <- specieslist(wkt="POLYGON((147.62 -42.83,147.60 -42.86,147.65 -42.87,147.70 -42.86,
  147.62 -42.83))",fq="rank:species")
```

```
x <- specieslist(wkt="POLYGON((145 -37,150 -37,150 -30,145 -30,145 -37))",fq="genus:Macropus")

## End(Not run)
```

`species_info` *Fetch a taxon profile given a scientific name or LSID (GUID)*

Description

Fetch a taxon profile given a scientific name or LSID (GUID)

Usage

```
species_info(scientificname, guid, verbose = ala_config()$verbose)
```

Arguments

`scientificname`	string: scientific name of the taxon of interest (species, genus, family etc)
`guid`	string: The Life Science Identifier of the taxon of interest
`verbose`	logical: show additional progress information? [default is set by ala_config()]

Value

A species profile in the form of a named list, each element of which is generally a data frame. An empty list is returned if no match is found for the supplied name or guid

References

Associated ALA web service: http://api.ala.org.au/#ws80

See Also

ala_config

Examples

```
## Not run:
 species_info("Grevillea humilis subsp. maritima")
 species_info(guid="http://id.biodiversity.org.au/node/apni/2890970")
 species_info("Alaba",verbose=TRUE)

## End(Not run)
```

taxinfo_download *Download taxonomic data*

Description

Download taxonomic data

Usage

```
taxinfo_download(query, fq, fields, verbose = ala_config()$verbose,
  use_data_table = TRUE)
```

Arguments

query	string: (optional) query of the form field:value (e.g. "genus:Macropus") or a free text search ("Alaba vibex")
fq	string: character string or vector of strings, specifying filters to be applied to the original query. These are of the form "INDEXEDFIELD:VALUE" e.g. "kingdom:Fungi". See `ala_fields("general",as_is=TRUE)` for all the fields that are queryable. NOTE that fq matches are case-sensitive, but sometimes the entries in the fields are not consistent in terms of case (e.g. kingdom names "Fungi" and "Plantae" but "ANIMALIA"). fq matches are ANDed by default (e.g. c("field1:abc","field2:def") will match records that have field1 value "abc" and field2 value "def"). To obtain OR behaviour, use the form c("field1:abc OR field2:def")
fields	string vector: (optional) a vector of field names to return. Note that the columns of the returned data frame are not guaranteed to retain the ordering of the field names given here. If not specified, a default list of fields will be returned. See `ala_fields("general",as_is=TRUE)` for valid field names. Use `fields="all"` to include all available fields
verbose	logical: show additional progress information? [default is set by ala_config()]
use_data_table	logical: if TRUE, attempt to read the data.csv file using the fread function from the data.table package. Requires data.table to be available. If this fails, or use_data_table is FALSE, then read.table will be used (which may be slower)

Value

data frame of results, containing one row per taxon, typically with name, guid, and taxonomic information. The columns returned will depend on the field requested

References

Associated ALA web service: http://api.ala.org.au/#ws2

See Also

ala_fields, ala_config

Examples

```
## Not run:
## simplest usage
x <- taxinfo_download("rk_genus:Macropus")

## Data for Fabaceae with specified fields
x <- taxinfo_download("rk_family:Fabaceae",fields=c("guid","parentGuid","rk_kingdom","rk_phylum",
  "rk_class","rk_order","rk_family","rk_genus","scientificName"))
# equivalent direct URL: http://bie.ala.org.au/ws/download?fields=guid,parentGuid,rk_kingdom,
#    rk_phylum,rk_class,rk_order,rk_family,rk_genus,scientificName&q=rk_family:Fabaceae

## End(Not run)
```

Index