

Package ‘suwo’

October 8, 2026

Type Package

Title Access Nature Media Repositories

Version 0.2.2

Maintainer Marcelo Araya-Salas <marcelo.araya@ucr.ac.cr>

Description Streamline searching, downloading and formatting of nature media files (e.g. audios, photos) from online repositories. The package offers functions for obtaining media metadata from online repositories, downloading associated media files and updating data sets with new records.

URL <https://docs.ropensci.org/suwo/>, <https://github.com/ropensci/suwo/>

BugReports <https://github.com/ropensci/suwo/issues/>

License GPL (>= 2)

Encoding UTF-8

Imports checkmate, cli, utils, rlang, httr2 (>= 1.1.0), jsonlite, lubridate, tools, RecordLinkage, leaflet

Suggests covr, testthat, knitr, rmarkdown, graphics, jpeg, fs, kableExtra, htmlwidgets, websocket, later, chromote

Depends R (>= 4.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Marcelo Araya-Salas [aut, cre] (ORCID: <<https://orcid.org/0000-0003-3594-619X>>),
Jorge Elizondo-Calvo [aut] (ORCID: <<https://orcid.org/0009-0004-5873-9585>>),
Alejandro Rico-Guevara [aut] (ORCID: <<https://orcid.org/0000-0003-4067-5312>>),
Hugo Gruson [rev] (ORCID: <<https://orcid.org/0000-0002-4094-1476>>),
Eric R Scott [rev] (ORCID: <<https://orcid.org/0000-0002-7430-7879>>)

Repository CRAN

Date/Publication 2026-10-08 08:30:02 UTC

Contents

access_wikiaves	2
download_media	5
find_duplicates	7
map_locations	9
merge_metadata	11
query_gbif	12
query_inaturalist	14
query_macaulay	16
query_wikiaves	19
query_xenocanto	21
remove_duplicates	25
update_metadata	27
Index	30

access_wikiaves	<i>Obtain WikiAves authentication cookies via a running Chrome or Chromium instance</i>
-----------------	---

Description

access_wikiaves authenticates with **WikiAves**, which sits behind Cloudflare’s bot protection and blocks plain http2 requests with an HTTP 403 and a "Just a moment..." challenge page. This function works around that by driving a real, visible Chromium-based browser (Google Chrome, Chromium, or another compatible build) through the Chrome DevTools Protocol (CDP): it launches the browser (or reuses one already running) pointed at WikiAves, waits for Cloudflare’s JavaScript challenge to clear, and then extracts the resulting cookies and the user agent string. The values are packed into a single character string.

By default (set_env = TRUE) this string is also stored directly in the wikiaves_cookies environment variable for the current R session, which is what query_wikiaves() reads from by default – so in the common case, calling access_wikiaves() once is enough to authenticate every subsequent query_wikiaves() call in the session without passing cookies explicitly. See Details.

This function works on Linux, macOS, and Windows. On all three platforms it will try to auto-detect a suitable browser and a writable temporary directory if chrome_bin and user_data_dir are left at their default (NULL); see Details.

Usage

```
access_wikiaves(  
  chrome_bin = NULL,  
  port = 9333,  
  url = "https://www.wikiaves.com.br/",  
  timeout = 10,  
  launch_wait = 15,  
  challenge_wait = 30,
```

```

    user_data_dir = NULL,
    set_env = TRUE
)

```

Arguments

chrome_bin	Character. Name or path of the browser executable to launch. Default NULL, which triggers auto-detection of a Chrome- or Chromium-based browser for the current operating system (see Details). Set this explicitly to override auto-detection or to point at a non-standard install location, e.g. <code>access_wikiaves("chromium")</code> on Linux, <code>"/Applications/Chromium.app/Contents/MacOS/Chromium"</code> on macOS, or <code>"C:/Program Files/Google/Chrome/Application/chrome.exe"</code> on Windows. Messages printed while the function runs refer to whichever browser <code>chrome_bin</code> resolves to, so it is easy to tell which one is in use.
port	Integer. The TCP port used for the browser's DevTools remote debugging protocol. Default 9333. If a browser instance is already listening on this port, it is reused instead of launching a new one.
url	Character. The WikiAves URL to navigate to and extract cookies from. Default <code>"https://www.wikiaves.com.br/"</code> . Changing this is only useful for testing against a different page on the same domain.
timeout	Numeric. Maximum time, in seconds, to wait for a single round-trip response (cookies, user agent, and page title) over the DevTools websocket connection. Default 10.
launch_wait	Numeric. Maximum time, in seconds, to wait for a newly launched browser process to open its DevTools debugging port. Default 15. Only relevant when the browser is not already running on port.
challenge_wait	Numeric. Maximum total time, in seconds, to keep polling the page while Cloudflare's "Just a moment..." challenge is showing. Default 30. If the challenge has not cleared within this window the function stops with an informative error; the person can solve the challenge manually in the visible browser window (e.g. clicking a checkbox) while this function is polling, and it will pick up the cleared page on the next check.
user_data_dir	Character. Filesystem path to the browser's user data directory used for this session. Default NULL, which resolves to a <code>"chrome-wikiaves-profile"</code> folder inside <code>tempdir()</code> – a location valid on Linux, macOS, and Windows alike. Reusing the same directory across calls lets the browser and Cloudflare recognize the profile as previously trusted, which often lets later calls clear the challenge automatically without manual interaction.
set_env	Logical. If TRUE (the default), also stores the returned cookies string in the <code>wikiaves_cookies</code> environment variable for the current R session via <code>Sys.setenv()</code> , so that <code>query_wikiaves()</code> 's default cookies argument picks it up automatically without needing to pass it explicitly. Set to FALSE to make the function a pure getter that only returns the cookies string without modifying the session's environment variables – useful if you want to manage the credentials yourself (e.g. pass them directly into a single <code>query_wikiaves()</code> call, store them somewhere else, or use a different session's variable).

Details

This function requires the **websocket** and **later** packages, which are listed under Suggests rather than Imports since this workflow is only needed for the WikiAves data source and depends on a local Chrome- or Chromium-based browser installation and a usable display (e.g. the DISPLAY environment variable on Linux). If either package is missing, in an interactive session the function asks for permission before installing them via `install.packages()`; declining stops the function with instructions to install manually. In a non-interactive session (e.g. within R CMD check, CI, or Rscript), the function cannot prompt and instead stops immediately with the same instructions.

Browser auto-detection. When `chrome_bin = NULL` (the default), the function first checks whether the **chromote** package happens to be installed and, if so, uses its exported `chromote::find_chrome()` to locate a browser – this respects the CHROMOTE_CHROME environment variable and **chromote**'s own per-OS search logic. **chromote** is not a dependency of this function; if it is not installed, or its detection does not find anything, the function falls back to its own candidate-path search:

- **Linux:** google-chrome, google-chrome-stable, chromium, and chromium-browser, in that order, checked via `Sys.which()`.
- **macOS:** /Applications/Google Chrome.app/Contents/MacOS/Google Chrome and /Applications/Chromium.app/Contents/MacOS/Chromium
- **Windows:** the chrome.exe install locations under Program Files and Program Files (x86), as well as a per-user AppData install location.

If none of these are found, the function stops with an error explaining how to set `chrome_bin` manually. Auto-detected paths have only been tested against typical default install locations; if your browser is installed somewhere non-standard, pass its path explicitly via `chrome_bin`.

On first use (or once the session has expired, which typically happens within a couple of hours), Cloudflare may present a visible interactive challenge in the launched browser window. Switching to that window and completing the challenge is a normal part of the workflow; this function polls for up to `challenge_wait` seconds so the person has time to do so before it gives up, and prints a message once the cookies have been successfully retrieved.

Automatic environment variable. By default (`set_env = TRUE`), the returned cookies string is also written to `Sys.setenv(wikiaves_cookies = ...)` before the function returns. Since `query_wikiaves()`'s `cookies` argument defaults to `Sys.getenv("wikiaves_cookies")`, this means a bare `access_wikiaves()` call is normally all that is needed to authenticate every `query_wikiaves()` call made afterwards in the same R session – there is no need to manually capture and pass the return value unless `set_env = FALSE` is used.

Value

A single JSON-encoded character string containing `cf_clearance`, `PHPSESSID`, and `browser_ua`. This is the format expected by the `cookies` argument of `query_wikiaves()`. The string is returned invisibly if `set_env = TRUE` (its side effect – setting `wikiaves_cookies` – is normally what matters in that case), and visibly if `set_env = FALSE`. Not expected to be modified by the user. Returns `invisible(NULL)` instead if no internet connection is available.

See Also

`query_wikiaves()`, which accepts the output of this function via its `cookies` argument.

Examples

```

if (interactive()) {
  # Simplest usage: launches (or reuses) a browser instance,
  # auto-detecting an appropriate Chrome/Chromium binary for the current
  # OS, solves the Cloudflare challenge if needed, and stores the
  # resulting cookies in the `wikiaves_cookies` environment variable
  # (set_env = TRUE by default) -- so query_wikiaves() picks them up
  # automatically without passing `cookies` explicitly:
  access_wikiaves()
  result <- query_wikiaves(species = "Procnias averano", format = "sound")

  # chrome_bin is the first argument, so a non-default browser binary can
  # be supplied positionally:
  access_wikiaves("chromium")

  # Using a non-default debugging port as well:
  access_wikiaves("chromium", port = 9444)

  # set_env = FALSE turns access_wikiaves() into a pure getter: nothing
  # is stored automatically, and the cookies string must be captured and
  # passed to query_wikiaves() (or elsewhere) manually:
  cookies_live <- access_wikiaves(set_env = FALSE)
  result <- query_wikiaves(
    species = "Procnias averano",
    format = "sound",
    cookies = cookies_live
  )
}

```

download_media

Download media files from repositories

Description

download_media downloads media files from online repositories.

Usage

```

download_media(
  metadata,
  path = ".",
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  verbose = getOption("suwo_verbose", TRUE),
  overwrite = FALSE,
  folder_by = NULL
)

```

Arguments

metadata	Data frame with the metadata of the media records to be mapped. Typically the output of one of the query functions in this package (e.g. query_gbif() , query_inaturalist() , etc.) or metadata formatting functions (e.g. merge_metadata() , remove_duplicates() , etc.).
path	Directory path where the output media files will be saved. By default files are saved into the current working directory (".").
cores	Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing). Can be set globally for the current R session via the "mc.cores" option (e.g. <code>options(mc.cores = 2)</code>). Note that some repositories might not support parallel queries from the same IP address as it might be identified as denial-of-service cyberattack.
pb	Logical argument to control if progress bar is shown. Default is TRUE. Can be set globally for the current R session via the "suwo_pb" option (<code>options(suwo_pb = TRUE)</code>). Not shown if only a few observations are found.
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).
overwrite	Logical. If TRUE, existing files (in "path") with the same name will be overwritten. Default is FALSE.
folder_by	Character string with the name of a character or factor column in the metadata data frame. If supplied the function will use the unique values in that column to create subfolders within "path" and the files will be downloaded into the corresponding folder. It can be used for instance to save files in subfolders by country or recordist. By default no subfolders are created and all files are saved in the path provided. Missing values (NAs) are saved in a folder called <code>paste0("unknown_", folder_by)</code> . Special characters that are not allowed in folder names will be modified or removed. If any of the folder names already exist in "path", they will be used as is.

Details

This function will take the output data frame of any of the "query_reponame()" functions and download the associated media files. The function will download all files into a single directory (argument "path"). File downloading process can be interrupted and resume later as long as the working directory is the same. Users only need to rerun the same function call. By default only the missing files will be downloaded when resuming. Can also be used on a updated query output (see [update_metadata\(\)](#)) to add the new media files to the existing media pool.

Value

Downloads media files into the supplied directory path ("path") and returns (invisibly) the input data frame with three additional columns: `downloaded_file_name` with the name of the downloaded file (if downloaded or already in the directory), `download_status` with the result of the download process for each file (either "saved", "overwritten", "already there (not downloaded)", or "failed"), and `file_size` with the size of the downloaded file in megabytes (MB). Note that the

column `file_size` can be used to further remove duplicated media using `find_duplicates()` and `remove_duplicates()` (e.g. if two files from different repositories have the same size and similar user name).

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

See Also

`query_gbif()`, `query_macaulay()`

Examples

```
a_zambiana <- query_inaturalist(species = "Amanita zambiana",
format = "image")

# run if query didnt fail
if (!is.null(a_zambiana)) {
  # download the first to files
  phae_anth_downl <- download_media(metadata = a_zambiana[1:2, ],
path = tempdir())
}
```

find_duplicates

Find duplicated entries in metadata

Description

`find_duplicates` detect possible duplicated entries from merged metadata from several repositories.

Usage

```
find_duplicates(
  metadata,
  sort = TRUE,
  criteria = paste("country > 0.8", "locality > 0.5", "user_name > 0.8", "time == 1",
    "date == 1", sep = " & "),
  verbose = getOption("suwo_verbose", TRUE)
)
```

Arguments

metadata	data frame obtained from combining the output metadata of two or more suwo query function using the <code>merge_metadata()</code> function. Data frames obtained from a single suwo query function can also be used but duplicates are not really expected within the same repository. The data frame must have the following columns: <code>user_name</code> , <code>locality</code> , <code>repository</code> , <code>country</code> , <code>format</code> , <code>time</code> , and <code>date</code> .
sort	Logical argument indicating if the output data frame should be sorted by the <code>duplicate_group</code> column added by this function. This will group all potential duplicates together in the output data frame. Default is <code>TRUE</code> .
criteria	A character string indicating the criteria to use to determine duplicates. By default, the criteria is set to <code>country > 0.8 & locality > 0.5 & user_name > 0.8 & time == 1 & date == 1</code> which means that two entries will be considered duplicates if they have a country similarity greater than 0.8, locality similarity greater than 0.5, <code>user_name</code> similarity greater than 0.8, and exact matches for time and date (similarities range from 0 to 1). These values have been found to work well in most cases. Users can modify this string to adjust the sensitivity of the duplicate detection based on their specific needs.
verbose	Logical argument that determines if text is shown in console. Default is <code>TRUE</code> . Can be set globally for the current R session via the <code>"suwo_verbose"</code> option (<code>options(suwo_verbose = TRUE)</code>).

Details

This function compares the information in the entries of a combined metadata data frame (typically the output of `merge_metadata()`) and labels those possible duplicates with a common index in a new column named `duplicate_group`. The comparison is based on the similarity of the following fields: `user_name`, `locality`, `time` and `country`. Only rows with no missing data for those fields will be considered. The function uses the `RecordLinkage` package to perform the a fuzzy matching comparison and identify potential duplicates based on predefined similarity thresholds (see argument `'criteria'`). The function only spots duplicates from different repositories and assumes those duplicates should have the same format and date. This function is useful for curating the data obtained by merging data from multiple sources, as the same observation might be recorded in different repositories. This is a common issue in citizen science repositories, where users might upload the same observation to different platforms. This can also occur as some repositories automatically share data with other repositories, particularly with GBIF. Note that the function does not remove any entries. The removal of duplicates can be done with the function `remove_duplicates()`. Further identification of duplicates can be done by checking the file size of the media (or including it as a criterion in the `'criteria'` argument). To obtain the file size, the media files need to be downloaded first with the function `download_media()` which returns the column `file_size` which can be used for this purpose.

Value

A data frame with the input data frame and an additional column named `duplicate_group` indicating potential duplicates with a common index. Entries without potential duplicates are labeled as NA in this new column.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

Examples

```
# get metadata from 2 repos
gb <- query_gbif(species = "Turdus rufiventris", format = "sound")
inat <- query_inaturalist(species = "Turdus rufiventris",
  format = "sound")

# run if queries didnt fail
if (!is.null(gb) && !is.null(inat)) {
  # combine metadata
  merged_metadata <- merge_metadata(inat, gb)

  # find duplicates
  label_dup_metadata <- find_duplicates(metadata = merged_metadata)
}
```

map_locations

Maps of media records

Description

map_locations creates maps to visualize the geographic spread of media records.

Usage

```
map_locations(
  metadata,
  cluster = FALSE,
  marker_color = NULL,
  by = "species",
  type = c("circles", "markers"),
  palette = function(n) grDevices::hcl.colors(n, "mako"),
  tags = c("repository", "key", "species", "date", "country", "locality", "user_name"),
  show_media = TRUE,
  popup_size = 1
)
```

Arguments

metadata	Data frame with the metadata of the media records to be mapped. Typically the output of one of the query functions in this package (e.g. query_gbif() , query_inaturalist() , etc.) or metadata formatting functions (e.g. merge_metadata() , remove_duplicates() , etc.). Note that only observations with geographic coordinates (i.e., non-missing values in the latitude and longitude columns) are displayed in the map.
----------	---

cluster	Logical to control if icons are clustered by locality. Default is FALSE. Only applies when type = "markers". When cluster = TRUE, markers that are close together will be clustered into a single marker that shows the number of observations in that cluster. Users can click on the cluster marker to zoom in and see the individual markers.
marker_color	Character vector with the color(s) to be used for the markers (when type = "markers". Possible values are "red", "darkred", "lightred", "orange", "beige", "green", "darkgreen", "lightgreen", "blue", "darkblue", "lightblue", "purple", "dark-purple", "pink", "cadetblue", "white", "gray", "lightgray", "black". Can be used to indicate the levels of a character or factor column with the argument "by". In such a case users must supplied as many colors as levels in the column.
by	Character string indicating the name of the column used to group observations for coloring. Default is "species". For type = "circles", this determines the color mapping and legend. For type = "markers", this determines how marker colors are assigned.
type	Character string indicating how observations are displayed. Options are "circles" (default) or "markers". Circles use a color palette and include a legend, while markers use colored icons and can be clustered.
palette	Function used to generate colors for circle markers when type = "circles". The function must take a single integer (n) and return n colors. By default it uses function(n) grDevices::hcl.colors(n, "mako"). Ignored when type = "markers".
tags	Character vector with the names of the columns in metadata to be shown in the popup. Default is c("repository", "key", "species", "date", "country", "locality", "user_name").
show_media	Logical indicating whether to display media files (audio, images, videos) in the popup windows. Default is TRUE.
popup_size	Numeric value that controls the size of the popups. Default is 1. Values greater than 1 will increase the size of the popups, while values between 0 and 1 will decrease it.

Details

The function uses the leaflet package to create interactive maps for visualizing the geographic spread of observations. Note that only observations with geographic coordinates are displayed. For each observation the function displays a marker in the map with a popup that shows the species name, country, locality, user name, and repository. The popup also includes an audio player for sound recordings, an image for photos and a video player for videos.

When multiple media files are associated with the same observation (i.e., identical values in the key column), they are grouped into a single popup. The first media item is displayed by default, and users can navigate through additional media using arrow buttons within the popup.

Users can zoom in and out of the map and click on the markers to see the popups. If cluster = TRUE, markers that are close together will be clustered into a single marker that shows the number of observations in that cluster. Users can click on the cluster marker to zoom in and see the individual markers. This function is useful for exploring the geographic distribution of media records and identifying patterns or gaps in the data.

Check the [leaflet package documentation](#) and the [leaflet.extras package](#) for more information on how to customize maps.

Value

An a leaflet interactive map object with the locations of the observations.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

Examples

```
if(interactive()){
# search in xeno-canto
e_hochs <- query_gbif(species = "Entoloma hochstetteri", format = "image")

# run if query didnt fail
if (!is.null(e_hochs)) {

  # create map
  map_locations(e_hochs)

  # create map without media
  map_locations(e_hochs, show_media = FALSE)
}
}
```

merge_metadata	<i>Merge metadata data frames</i>
----------------	-----------------------------------

Description

merge_metadata merges metadata data frames from suwo queries.

Usage

```
merge_metadata(..., check_columns = TRUE)
```

Arguments

...	two or more data frames (each one as a separate entry) referring to the metadata obtained from suwo query functions (query_x()). Alternatively, a single list of data frames can be provided. The name provided for each data frame (either as individual data frames or in a list) will be used as label in the source column in the output data frame.
check_columns	Logical argument indicating if the function should check that all input data frames have the required basic columns. Default is TRUE.

Details

This function combines metadata from multiple sources (e.g. WikiAves and xeno-canto) into a single data frame for easier analysis and comparison. Each input data frame must be obtained from one of the suwo query functions (e.g., query_wikiaves(), query_xenocanto(), etc.) with raw_data = FALSE.

Value

A single data frame with the data from all input data frames combined and with an additional column named source indicating the original data frame from which each row originated. The column source will contain the name provided for each data frame (either as individual data frames or in a list). If no names were provided, the object names will be used instead.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

Examples

```
# get metadata from 2 repos
wa <- query_wikiaves(species = "Glaucis dohrnii", format = "sound")
gb <- query_gbif(species = "Glaucis dohrnii", format = "sound")

# run if queries didnt fail
if (!is.null(wa) && !is.null(gb)) {
  # combine metadata using single data frames
  merged_mt <- merge_metadata(wa, gb)

  # combine metadata using named single data frames
  merged_mt <- merge_metadata(wikiaves = wa, gbif = gb)

  # combine metadata using a list of data frames
  mt_list <- list(wikiaves = wa, gbif = gb)
  merged_mt <- merge_metadata(mt_list)
}
```

query_gbif

Access 'gbif' media file metadata

Description

query_gbif searches for metadata from **GBIF**.

Usage

```
query_gbif(
  species = getOption("suwo_species"),
  format = getOption("suwo_format", c("image", "sound", "video", "interactive resource")),
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  verbose = getOption("suwo_verbose", TRUE),
  dataset = NULL,
  all_data = getOption("suwo_all_data", FALSE),
  raw_data = getOption("suwo_raw_data", FALSE)
)
```

Arguments

species	Character string with the scientific name of a species in the format: "Genus epithet". Required. Can be set globally for the current R session via the "suwo_species" option (e.g. <code>options(suwo_species = "Hypsiboas rufitelus")</code>).
format	Character vector with the media format to query for. Options are 'sound', 'image', 'video' and 'interactive resource'. Can be set globally for the current R session via the "suwo_format" option (e.g. <code>options(suwo_format = "image")</code>). Required.
cores	Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing). Can be set globally for the current R session via the "mc.cores" option (e.g. <code>options(mc.cores = 2)</code>). Note that some repositories might not support parallel queries from the same IP address as it might be identified as denial-of-service cyberattack.
pb	Logical argument to control if progress bar is shown. Default is TRUE. Can be set globally for the current R session via the "suwo_pb" option (<code>options(suwo_pb = TRUE)</code>). Not shown if only a few observations are found.
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).
dataset	The name of a specific dataset in which to focus the query (by default it searches across all available datasets). Users can check available dataset names by downloading this csv file https://api.gbif.org/v1/dataset/search/export?format=CSV& . See https://www.gbif.org/dataset/search?q= for more details.
all_data	Logical argument that determines if all data available from database is shown in the results of search. Default is FALSE. Can be set globally for the current R session via the "suwo_all_data" option (<code>options(suwo_all_data = TRUE)</code>).
raw_data	Logical argument that determines if the raw data from the repository is returned (e.g. without any manipulation). Default is FALSE. Can be set globally for the current R session via the "suwo_raw_data" option (<code>options(suwo_raw_data = TRUE)</code>). If TRUE <code>all_data</code> is set to TRUE internally. Useful for developers, or if users suspect that some data is mishandled during processing (i.e. date

information is lost). Note that the metadata obtained when `raw_data = TRUE` is not standardized, so most suwo functions for downstream steps will not work on them.

Details

This function queries for species observation info in the open-access online repository **GBIF**. GBIF (the Global Biodiversity Information Facility) is an international network and data infrastructure funded by the world's governments and aimed at providing open access to data about all types of life on Earth. Note that some of the records returned by this function could be duplicates of records returned by other suwo functions (e.g., `query_inaturalist()`).

Value

The function returns a data frame with the metadata of the media files matching the search criteria. If `all_data = TRUE`, all metadata fields (columns) are returned. If `raw_data = TRUE`, the raw data as obtained from the repository is returned (without any formatting).

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

References

GBIF.org (2024), GBIF Home Page. Available from: <https://www.gbif.org/>

See Also

`query_inaturalist()`

Examples

```
# search dink frog sound files
d_diastema <- query_gbif(species = "Diasporus diastema", format = "sound")
```

query_inaturalist	Access 'iNaturalist' media file metadata
-------------------	--

Description

`query_inaturalist` searches for metadata from **iNaturalist**.

Usage

```
query_inaturalist(
  species = getOption("suwo_species"),
  format = getOption("suwo_format", c("image", "sound")),
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  verbose = getOption("suwo_verbose", TRUE),
  all_data = getOption("suwo_all_data", FALSE),
  raw_data = getOption("suwo_raw_data", FALSE),
  identified = FALSE,
  verifiable = FALSE
)
```

Arguments

species	Character string with the scientific name of a species in the format: "Genus epithet". Required. Can be set globally for the current R session via the "suwo_species" option (e.g. <code>options(suwo_species = "Hypsiboas rufitellus")</code>).
format	Character vector with the media format to query for. Currently 'image' and 'sound' are available. Can be set globally for the current R session via the "suwo_format" option (e.g. <code>options(suwo_format = "image")</code>). Required.
cores	Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing). Can be set globally for the current R session via the "mc.cores" option (e.g. <code>options(mc.cores = 2)</code>). Note that some repositories might not support parallel queries from the same IP address as it might be identified as denial-of-service cyberattack.
pb	Logical argument to control if progress bar is shown. Default is TRUE. Can be set globally for the current R session via the "suwo_pb" option (<code>options(suwo_pb = TRUE)</code>). Not shown if only a few observations are found.
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).
all_data	Logical argument that determines if all data available from database is shown in the results of search. Default is TRUE.
raw_data	Logical argument that determines if the raw data from the repository is returned (e.g. without any manipulation). Default is FALSE. Can be set globally for the current R session via the "suwo_raw_data" option (<code>options(suwo_raw_data = TRUE)</code>). If TRUE all_data is set to TRUE internally. Useful for developers, or if users suspect that some data is mishandled during processing (i.e. date information is lost). Note that the metadata obtained when raw_data = TRUE is not standardized, so most suwo functions for downstream steps will not work on them.
identified	Logical argument to define if search results are categorized as identified by inaturalist.
verifiable	Logical argument to define if search results are categorized as verifiable by inaturalist.

Details

This function queries for species observation info in the open-access online repository [iNaturalist](#). iNaturalist is a free, crowdsourced online platform for nature enthusiasts to document and identify plants, animals, fungi, and other organisms in the wild. Note that iNaturalist observations do not include a 'country' field.

Value

The function returns a data frame with the metadata of the media files matching the search criteria. If `all_data = TRUE`, all metadata fields (columns) are returned. If `raw_data = TRUE`, the raw data as obtained from the repository is returned (without any formatting).

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

References

iNaturalist. Available from <https://www.inaturalist.org>. (Accessed on 10-02-2026)

Examples

```
# search Bleeding Tooth mushroom images
```

query_macaulay

Searches for media files in the Macaulay Library

Description

query_macaulay searches for metadata from [Macaulay library](#).

Usage

```
query_macaulay(
  species = getOption("suwo_species"),
  taxon_code = NULL,
  format = getOption("suwo_format", c("image", "sound", "video")),
  verbose = getOption("suwo_verbose", TRUE),
  all_data = getOption("suwo_all_data", FALSE),
  raw_data = getOption("suwo_raw_data", FALSE),
  path = ".",
  files = NULL,
  dates = NULL,
  taxon_code_info = ml_taxon_code
)
```

Arguments

species	Character string with the scientific name of a species in the format: "Genus epithet". Required. Can be set globally for the current R session via the "suwo_species" option (e.g. <code>options(suwo_species = "Hypsiboas rufitelus")</code>).
taxon_code	Optional character string with the Macaulay Library taxon code (see vignette for more details). If provided, 'species' is ignored.
format	Character vector with the media format to query for. Options are 'sound', 'image' or 'video'. Can be set globally for the current R session via the "suwo_format" option (e.g. <code>options(suwo_format = "image")</code>). Required.
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).
all_data	Logical argument that determines if all data available from database is shown in the results of search. Default is FALSE. Can be set globally for the current R session via the "suwo_all_data" option (<code>options(suwo_all_data = TRUE)</code>).
raw_data	Logical argument that determines if the raw data from the repository is returned (e.g. without any manipulation). Default is FALSE. Can be set globally for the current R session via the "suwo_raw_data" option (<code>options(suwo_raw_data = TRUE)</code>). If TRUE <code>all_data</code> is set to TRUE internally. Useful for developers, or if users suspect that some data is mishandled during processing (i.e. date information is lost). Note that the metadata obtained when <code>raw_data = TRUE</code> is not standardized, so most suwo functions for downstream steps will not work on them.
path	Directory path where the .csv file will be saved. By default it is saved into the current working directory (".").
files	Optional character vector with the name(s) of the .csv file(s) to read. If provided, the function will import the data from the .csv files instead of opening the Macaulay Library search page in a browser ('species' is ignored if supplied).
dates	Optional numeric vector with years to split the search. If provided, the function will perform separate queries for each date range (between consecutive date values) and combine the results. Useful for queries that return large number of results (i.e. > 10000 results limit). For example, to search for the species between 2010 to 2020 and between 2021 to 2025 use <code>dates = c(2010, 2020, 2025)</code> . If years contain decimals searches will be split by months within years as well.
taxon_code_info	Data frame containing the taxon code information. By default the function will use the internal data frame "ml_taxon_code" included as example data in the package. This object contains the data from the October-2025 eBird taxonomy (downloaded from https://www.birds.cornell.edu/clementschecklist). If new versions of the list become available it will be updated in new package versions. However, if users need to update it they can download the new list version, read it in R as a data frame and provide it to the function through this argument.

Details

This function queries for species observation info in the [Macaulay library](#) online repository and returns the metadata of media files matching the query. The Macaulay Library is the world's largest repository of digital media (audio, photo, and video) of wildlife (mostly birds but also other vertebrates and invertebrates), and their habitats. The archive hosts more than 77 million images, 3 million sound recordings, and 350k videos, from more than 80k contributors, and is integrated with eBird, the world's largest biodiversity dataset. For bird species the species name must be valid according to the Macaulay Library taxonomy (which follows the Clements checklist). For non-bird species users must use the argument `taxon_code`. The species taxon code can be found by running a search at the [Macaulay Library's search page](#) and checking the URL of the species page. For instance, the URL when searching for jaguar (*Panthera onca*) is 'https://search.macaulaylibrary.org/catalog?taxonCode=t-11032765' so the taxon code is "t-11032765". If `all_data = TRUE`, all metadata fields (columns) are returned. If `raw_data = TRUE`, the raw data as obtained from the repository is returned (without any formatting). Here are some instructions for using this function properly:

- Valid bird species names can be checked at `suwo::ml_taxon_code$SCI_NAME`.
- Users must save the .csv file manually
- If the file is saved overwriting a pre-existing file (i.e. same file name) the function will not detect it
- A maximum of 10000 records per query can be returned, but this can be bypassed by using the `dates` argument to split the search into smaller date ranges
- Users must log in to the Macaulay Library/eBird account in order to access large batches of observations

Value

This is an interactive function which opens a browser window to the Macaulay Library's search page, where the user must download a .csv file with the metadata. The function then reads the .csv file and returns a data frame with the metadata. The function can also import previously downloaded metadata (in csv format) with the argument `files`.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

References

- Scholes III, Ph.D. E (2015). Macaulay Library Audio and Video Collection. Cornell Lab of Ornithology. Occurrence dataset <https://doi.org/10.15468/ckcdpy> accessed via GBIF.org on 2024-05-09.
- Clements, J. F., P. C. Rasmussen, T. S. Schulenberg, M. J. Iliff, J. A. Gerbracht, D. Lepage, A. Spencer, S. M. Billerman, B. L. Sullivan, M. Smith, and C. L. Wood. 2025. The eBird/Clements checklist of Birds of the World: v2025. Downloaded from <https://www.birds.cornell.edu/clementschecklist/download/>

Examples

```
if (interactive()){
# query sounds
```

```

tur_ili <- query_macaulay(species = "Turdus iliacus", format = "sound",
path = tempdir())

# test a query with more than 10000 results paging by date
# this example splits by entire year intervals
cal_cos <- query_macaulay(species = "Calypte costae", format = "image",
path = tempdir(), dates = c(1976, 2019, 2022, 2024, 2025, 2026))

# this example splits by year-month intervals (as dates have decimals)
cal_cos <- query_macaulay(species = "Calypte costae", format = "image",
path = tempdir(), dates = seq(2020, 2026, length.out = 10))

## update clement list (note that this is actually the same list used in the
# current 'suwo' version, just for the sake of the example)

# url to the clements list version october 2024
# (split so it is not truncated by CRAN)
clements_url <- paste0(
  "https://www.birds.cornell.edu/clementschecklist/wp-content/uploads/2024/10",
  "/Clements-v2024-October-2024-rev.csv"
)

# read list from url
new_clements <- utils::read.csv(clements_url)

# provide "updated" clements list to query_macaulay()
tur_ili2 <- query_macaulay(species = "Turdus iliacus", format = "sound",
  taxon_code_info = new_clements, path = tempdir())

# query using taxon code
# this is the URL when querying jaguars:
# https://search.macaulaylibrary.org/catalog?taxonCode=t-11032765
p_onca <- query_macaulay(taxon_code = "t-11032765", format = "image")
}

```

query_wikiaves

Query WikiAves for bird media metadata

Description

query_wikiaves searches for metadata from **WikiAves**, a Brazilian citizen-science platform and the largest birdwatching community in the country. It functions as a collaborative, interactive encyclopedia of Brazilian birds, where users contribute georeferenced photographs and sound recordings that build a large database used for research and conservation. query_wikiaves returns metadata for image or sound recordings matching a given species, including download links, locality, author, and verification status.

WikiAves sits behind Cloudflare's bot protection, so this function requires authentication, which is done with [access_wikiaves\(\)](#) (see Examples).

Usage

```
query_wikiaves(
  species = getOption("suwo_species"),
  format = getOption("suwo_format", c("image", "sound")),
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  verbose = getOption("suwo_verbose", TRUE),
  all_data = getOption("suwo_all_data", FALSE),
  raw_data = getOption("suwo_raw_data", FALSE),
  cookies = Sys.getenv("wikiaves_cookies")
)
```

Arguments

species	Character string giving the scientific name of the species to search for (e.g. "Procnias averano"). Defaults to <code>getOption("suwo_species")</code> .
format	Character string, either "image" or "sound", indicating which type of media to query. Defaults to <code>getOption("suwo_format", c("image", "sound"))</code> .
cores	Numeric. Number of cores to use for parallel processing of paginated results. Defaults to <code>getOption("suwo_cores", 1)</code> .
pb	Logical. Whether to show a progress bar. Defaults to <code>getOption("suwo_pb", TRUE)</code> .
verbose	Logical. Whether to print progress and error messages. Defaults to <code>getOption("suwo_verbose", TRUE)</code> .
all_data	Logical. Whether to return all available columns rather than the standard suwo output columns. Defaults to <code>getOption("suwo_all_data", FALSE)</code> .
raw_data	Logical. Whether to return the raw, unformatted query output instead of the standardized suwo output. Defaults to <code>getOption("suwo_raw_data", FALSE)</code> .
cookies	Single character string of WikiAves authentication credentials, used to get past Cloudflare's bot protection on every request this function makes. It is generated by <code>access_wikiaves()</code> and saved as an environmental variable (<code>Sys.getenv("wikiaves_cookies")</code>), which is read by default by <code>query_wikiaves</code> .

Details

WikiAves sits behind Cloudflare's bot protection, so cookies must be a valid, non-expired credentials string – see **Do I need to obtain new credentials before every call?** below.

Do I need to obtain new credentials before every call? No – only once per Cloudflare session. The `cf_clearance` cookie returned by `access_wikiaves()` is typically valid for roughly an hour (Cloudflare does not publish an exact figure, and the duration can vary); after that, requests made with it will start failing again with an HTTP 403, and `access_wikiaves()` simply needs to be called again to obtain a fresh set of credentials.

A few practical notes on when to refresh credentials:

- If `query_wikiaves` is paging through a species with a large number of recordings, the session may expire partway through a single call – if that happens, call `access_wikiaves()` again and retry.

- If some time has passed since the cookies were last obtained, refresh them before querying rather than waiting for a failure.
- `access_wikiaves()` reuses the same browser profile directory by default, so repeat calls are usually fast and do not require solving the Cloudflare challenge by hand each time.

If valid cookies are not supplied, `query_wikiaves` fails gracefully with an explanatory message and returns `invisible(NULL)` rather than throwing an error.

Value

A data frame of WikiAves observations matching species and format, with standardized suwo columns (or all available columns if `all_data = TRUE`, or the raw query output if `raw_data = TRUE`). Returns `invisible(NULL)` if the species is not found, no matching records exist, or the query otherwise fails.

See Also

`access_wikiaves()`, which generates the cookies argument this function requires.

Examples

```
if (interactive()) {
  # Obtain fresh authentication cookies (only needs to be re-run once the
  # cookies expire, roughly every hour):
  cookies_live <- access_wikiaves()

  # Query sound recordings for a species:
  result <- query_wikiaves(
    species = "Procnias averano",
    format = "sound",
    cookies = cookies_live
  )

  # Query image records instead:
  result_images <- query_wikiaves(
    species = "Procnias averano",
    format = "image",
    cookies = cookies_live
  )
}
```

query_xenocanto

Access 'Xeno-Canto' recording metadata

Description

`query_xenocanto` searches for metadata from **Xeno-Canto**.

Usage

```
query_xenocanto(
  species = getOption("suwo_species"),
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  verbose = getOption("suwo_verbose", TRUE),
  all_data = getOption("suwo_all_data", FALSE),
  raw_data = getOption("suwo_raw_data", FALSE),
  api_key = Sys.getenv("xc_api_key")
)
```

Arguments

species	Character string with the scientific name of a species in the format: "Genus epithet". Required. Can be set globally for the current R session via the "suwo_species" option (e.g. <code>options(suwo_species = "Hypsiboas rufitelus")</code>). Alternatively, a character string containing additional tags that follows the Xeno-Canto advanced query syntax can be provided. Tags are of the form 'tag:searchterm'. For instance, 'type:"song"' will search for recordings where the sound type contains 'song'. Multiple tags can be provided (e.g., '"cnt:"belize" type:"song"'). This includes Xeno-Canto's annotation-specific search tags (e.g. <code>ann_sp</code> , <code>ann_type</code> , <code>ann_gen</code> , <code>ann_frq_low</code> , <code>ann_frq_high</code> ; see Xeno-Canto's search help linked below) to filter for recordings that have a matching annotation – note this only filters <i>which recordings</i> are returned; the actual annotation-level data for any returned recording is available regardless of whether annotation tags were used in the search (see Details). See examples down below and check Xeno-Canto's search help for a full description.
cores	Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing). Can be set globally for the current R session via the "mc.cores" option (e.g. <code>options(mc.cores = 2)</code>). Note that some repositories might not support parallel queries from the same IP address as it might be identified as denial-of-service cyberattack.
pb	Logical argument to control if progress bar is shown. Default is TRUE. Can be set globally for the current R session via the "suwo_pb" option (<code>options(suwo_pb = TRUE)</code>). Not shown if only a few observations are found.
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).
all_data	Logical argument that determines if all data available from database is shown in the results of search. Default is FALSE. Can be set globally for the current R session via the "suwo_all_data" option (<code>options(suwo_all_data = TRUE)</code>).
raw_data	Logical argument that determines if the raw data from the repository is returned (e.g. without any manipulation). Default is FALSE. Can be set globally for the current R session via the "suwo_raw_data" option (<code>options(suwo_raw_data = TRUE)</code>). If TRUE <code>all_data</code> is set to TRUE internally. Useful for developers, or if users suspect that some data is mishandled during processing (i.e. date

information is lost). Note that the metadata obtained when `raw_data = TRUE` is not standardized, so most `suwo` functions for downstream steps will not work on them.

`api_key` Character string referring to the key assigned by Xeno-Canto as authorization for searches. Get yours at <https://xeno-canto.org/account>. Required. Avoid setting your API key directly in the function call to prevent exposing it in your code. Instead, set it as an environment variable (e.g., in your `.Renv` file using `Sys.setenv(xc_api_key = "your_key_here")`) named `'xc_api_key'`, so it can be accessed securely using `Sys.getenv("xc_api_key")`.

Details

This function queries metadata for animal sound recordings in the open-access online repository **Xeno-Canto**. **Xeno-Canto** hosts sound recordings of birds, frogs, non-marine mammals and grasshoppers. Complex queries can be constructed using the **Xeno-Canto** advanced query syntax (see examples).

Annotations. Some Xeno-Canto recordings have one or more annotations attached: individually marked sound segments within the recording, each with their own scientific name, annotator, start/end time (in seconds), frequency range (in Hz), sound type, sex, life stage, and remarks. This is different from (and more granular than) the recording-level metadata returned in the main result – a single recording can have several annotations, each describing a different segment of that one sound file.

Xeno-Canto includes this annotation data in the same API response used to fetch ordinary recording metadata, so retrieving it costs no extra requests and is always extracted (there is no separate argument to turn this on or off). When present, it is available as a data frame via `attr(result, "annotations")`, with one row per annotation; a message reports how many annotations were found and confirms they were added to this attribute. For reference, each annotation row also includes `key` (the Xeno-Canto ID of the recording it belongs to, matching the `key` column of the main result – use this to join back to the main result if needed), `species` (the species identified in that specific annotated segment – usually, but not necessarily always, the same as the parent recording's overall species; it can differ for an annotated background call of a different species), plus `file_url` and `observation_url` links to the parent recording.

To search specifically for recordings that have annotations matching certain criteria (rather than just inspecting whatever annotations happen to be attached to whatever recordings a search returns), use Xeno-Canto's `ann_*` search tags directly in `species`, e.g. `species = 'ann_sp: "Turdus migratorius" ann_type: "call"'`.

Value

The function returns a data frame with the metadata of the media files matching the search criteria. If `all_data = TRUE`, all metadata fields (columns) are returned. If `raw_data = TRUE`, the raw data as obtained from the repository is returned (without any formatting).

If any of the matching recordings have Xeno-Canto annotations attached (individually annotated sound segments within a recording, distinct from the recording-level metadata above), those are included as a data frame attached to the result via `attr(result, "annotations")` – see Details. If no matching recordings have any annotations, this attribute is not set (`attr(result, "annotations")` is `NULL`).

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

References

Planqué, Bob, & Willem-Pier Vellinga. 2008. Xeno-canto: a 21st-century way to appreciate Neotropical bird song. *Neotrop. Birding* 3: 17-23.

See Also

[query_gbif\(\)](#), [query_wikiaves\(\)](#), [query_inaturalist\(\)](#), [download_media\(\)](#)

Examples

```
if (interactive()){
# An API key is required. Get yours at https://xeno-canto.org/account.
# run this in the console but dont save it in a script
Sys.setenv(xc_api_key = "YOUR_API_KEY_HERE")

# Simple search for a species
p_anth <- query_xenocanto(species = "Phaethornis anthophilus")

# Search for same species and add specify country
p_anth_cr <- query_xenocanto(
species = 'sp:"Phaethornis anthophilus" cnt:"Panama"',
raw_data = TRUE)

# Search for female songs of a species
femsong <- query_xenocanto(
species = 'sp:"Thryothorus ludovicianus" type:"song" type:"female"')

# Any annotations attached to the returned recordings are available as
# an attribute, regardless of the search used:
poospiza <- query_xenocanto(species = "Poospiza hispaniolensis")
annotations <- attr(poospiza, "annotations")
annotations

# each annotation row includes `key` and `species` for easy reference
# back to the parent recording, e.g. to join with the main result:
if (!is.null(annotations)) {
  merge(annotations, poospiza, by = "key")
}

# Searching with an annotation-specific tag filters which recordings
# are returned, but does not change how annotation data is retrieved:
ann_search <- query_xenocanto(species = 'ann_sp:"Poospiza hispaniolensis"')
}
```

remove_duplicates	<i>Remove duplicated media records</i>
-------------------	--

Description

remove_duplicates removes duplicated media records.

Usage

```
remove_duplicates(
  metadata,
  same_repo = FALSE,
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  repo_priority = c("Xeno-Canto", "GBIF", "iNaturalist", "Macaulay Library", "WikiAves"),
  verbose = getOption("suwo_verbose", TRUE)
)
```

Arguments

metadata	data frame obtained with the function <code>find_duplicates()</code> . The data frame must have the column 'duplicate_group' returned by <code>find_duplicates()</code> .
same_repo	Logical argument indicating if observations labeled as duplicates that belong to the same repository should be removed. Default is FALSE. If TRUE, only one of the duplicated observations from the same repository will be retained in the output data frame. This is useful as it can be expected that observations from the same repository are not true duplicates (e.g. different recordings uploaded to Xeno-Canto with the same date, time and location by the same user), but rather have not been documented with enough precision to be told apart.
cores	Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing). Can be set globally for the current R session via the "mc.cores" option (e.g. <code>options(mc.cores = 2)</code>). Note that some repositories might not support parallel queries from the same IP address as it might be identified as denial-of-service cyberattack.
pb	Logical argument to control if progress bar is shown. Default is TRUE. Can be set globally for the current R session via the "suwo_pb" option (<code>options(suwo_pb = TRUE)</code>). Not shown if only a few observations are found.
repo_priority	Character vector indicating the priority of repositories when selecting which observation to retain when duplicates are found. Default is <code>c("Xeno-Canto", "GBIF", "iNaturalist", "Macaulay Library", "Wikiaves")</code> , which gives priority to repositories in which media downloading is more straightforward (Xeno-Canto and GBIF).
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).

Details

When compiling data from multiple repositories, duplicated media records are a common issue, particularly for sound recordings. These duplicates occur both through data sharing between repositories like Xeno-Canto and GBIF, and when users upload the same file to multiple platforms. In such cases those multiple observations seem to refer to the same media file and therefore, only one copy is needed. This function removes duplicate observations identified with the function `find_duplicates()`. When duplicates are found, one observation from each group of duplicates is retained in the output data frame. However, if multiple observations from the same repository are labeled as duplicates, by default (`same_repo = FALSE`) all of them are retained in the output data frame. This is useful as it can be expected that observations from the same repository are not true duplicates (e.g. different recordings uploaded to Xeno-Canto with the same date, time and location by the same user), but rather have not been documented with enough precision to be told apart. This behavior can be modified. If `same_repo = TRUE`, only one of the duplicated observations from the same repository will be retained in the output data frame. The function will give priority to repositories in which media downloading is more straightforward (Xeno-Canto and GBIF), but this can be modified with the argument `'repo_priority'`.

Value

A single data frame with a subset of the `'metadata'` with those observations that were determined not to be duplicates.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

See Also

`find_duplicates()`, `merge_metadata()`

Examples

```
if(interactive()){
# get metadata from 2 repos
gb <- query_gbif(species = "Turdus rufiventris", format = "sound")

key <- "YOUR XENO CANTO API KEY"
xc <- query_xenocanto(species = "Turdus rufiventris", api_key = key)

# combine metadata
merged_metadata <- merge_metadata(xc, gb)

# find duplicates
label_dup_metadata <- find_duplicates(metadata = merged_metadata)

# remove duplicates
dedup_metadata <- remove_duplicates(label_dup_metadata)
}
```

update_metadata	<i>Update metadata</i>
-----------------	------------------------

Description

update_metadata update metadata from previous queries.

Usage

```
update_metadata(
  metadata,
  path = ".",
  cores = getOption("suwo_cores", 1),
  pb = getOption("suwo_pb", TRUE),
  verbose = getOption("suwo_verbose", TRUE),
  api_key = NULL,
  dates = NULL
)
```

Arguments

metadata	Data frame with the metadata of media records. Typically the output of one of the query functions in this package (e.g. query_gbif() , query_inaturalist() , etc.).
path	Directory path where the .csv file will be saved. Only applicable for query_macaulay() query results. By default it is saved into the current working directory (".").
cores	Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing). Can be set globally for the current R session via the "mc.cores" option (e.g. <code>options(mc.cores = 2)</code>). Note that some repositories might not support parallel queries from the same IP address as it might be identified as denial-of-service cyberattack.
pb	Logical argument to control if progress bar is shown. Default is TRUE. Can be set globally for the current R session via the "suwo_pb" option (<code>options(suwo_pb = TRUE)</code>). Not shown if only a few observations are found.
verbose	Logical argument that determines if text is shown in console. Default is TRUE. Can be set globally for the current R session via the "suwo_verbose" option (<code>options(suwo_verbose = TRUE)</code>).
api_key	Character string referring to the key assigned by Xeno-Canto as authorization for searches. Get yours at https://xeno-canto.org/account . Only needed if the input metadata comes from query_xenocanto() .
dates	Optional numeric vector with years to split the search. If provided, the function will perform separate queries for each date range (between consecutive date values) and combine the results. Useful for queries that return large number of results (i.e. > 10000 results limit). For example, to search for the species

between 2010 to 2020 and between 2021 to 2025 use `dates = c(2010, 2020, 2025)`. If years contain decimals searches will be split by months within years as well. Only needed if the input metadata comes from `query_macaulay()`.

Details

This function updates the metadata from a previous query to add entries found in the source repository. **All observations must belong to the same repository** (but see examples for code to update metadata from multiple repositories). The function adds the column `new_entry` which labels those entries that are new (i.e., not present in the input metadata). The input data frame must have been obtained from any of the query functions with the argument `raw_data = FALSE`. The function uses the same query species and format as in the original query. If no new entries are found, the function returns the original metadata and prints a message. If some old entries are not returned in the new query they are still retained. The function assumes that no new files are added to existing repository entries. The value of `all_data` (an argument common to all query functions) is inferred from the columns present in metadata. If columns beyond the standard output are detected, the function assumes `all_data = TRUE`. Columns added during processing by any `suwo` function ("`source`", "`new_entry`", "`downloaded_file_name`", "`download_status`", "`file_size`", "`duplicate_group`") are ignored to prevent incorrect inference.

Value

returns a data frame similar to the input 'metadata' with new data appended.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

Examples

```
# query metadata
a_gioiosa <- query_gbif(species = "Amanita gioiosa", format = "image")

# run if query didnt fail
if (!is.null(a_gioiosa)) {

  # remove the key with more observations
  sub_a_gioiosa <-
  a_gioiosa[a_gioiosa$key != names(which.max(table(a_gioiosa$key))), ]

  # update
  up_a_gioiosa <- update_metadata(metadata = sub_a_gioiosa)

  # check number of rows is the same (e.g. it has been updated)
  nrow(up_a_gioiosa) == nrow(a_gioiosa)

  # example multi repository update

  a_orientigemmata <- query_inaturalist(species = "Amanita orientigemmata",
  format = "image")
```

```
#remove the key with more observations
sub_a_orientigemmata <-
a_orientigemmata[a_orientigemmata$key !=
names(which.max(table(a_orientigemmata$key))), ]

# merge both metadata
sub_amanitas <- merge_metadata(sub_a_gioiosa, sub_a_orientigemmata)

# split by repository and update separately
up_amanitas_list <-
lapply(split(sub_amanitas, sub_amanitas$repository), update_metadata)

# merge updated metadata
up_amanitas <- do.call(merge_metadata, up_amanitas_list)

# check number of rows is the same (e.g. it has been updated)
nrow(up_amanitas) == nrow(a_gioiosa) + nrow(a_orientigemmata)

}
```

Index

`access_wikiaves`, [2](#)
`access_wikiaves()`, [19–21](#)

`download_media`, [5](#)
`download_media()`, [8](#), [24](#)

`find_duplicates`, [7](#)
`find_duplicates()`, [7](#), [25](#), [26](#)

`map_locations`, [9](#)
`merge_metadata`, [11](#)
`merge_metadata()`, [6](#), [8](#), [9](#), [26](#)

`query_gbif`, [12](#)
`query_gbif()`, [6](#), [7](#), [9](#), [24](#), [27](#)
`query_inaturalist`, [14](#)
`query_inaturalist()`, [6](#), [9](#), [14](#), [24](#), [27](#)
`query_macaulay`, [16](#)
`query_macaulay()`, [7](#), [27](#), [28](#)
`query_wikiaves`, [19](#)
`query_wikiaves()`, [2–4](#), [24](#)
`query_xenocanto`, [21](#)
`query_xenocanto()`, [27](#)

`remove_duplicates`, [25](#)
`remove_duplicates()`, [6–9](#)

`update_metadata`, [27](#)
`update_metadata()`, [6](#)