Package: lifeR (via r-universe)

September 8, 2024

Type Package

Title Identify Sites for Your Bird List

Version 1.0.2

Description A suite of tools to use the 'eBird' database
(<https://ebird.org/home/>) and APIs to compare users' species
lists to recent observations and create a report of the top
sites to visit to see new species.

License BSD_2_clause + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.2.0)

Imports curl (>= 4.3), dplyr (>= 1.0.2), ggplot2 (>= 3.4.4), jsonlite
(>= 1.7.0), knitr (>= 1.31), maptiles (>= 0.6.1), readr (>=
1.4.0), rmarkdown (>= 2.7), stringr (>= 1.4.0), terra (>=
1.7-55), tidyterra (>= 0.5.0)

RoxygenNote 7.3.1

Suggests testthat

VignetteBuilder knitr

URL https://jcoliver.github.io/lifeR/,
https://github.com/jcoliver/lifeR/

Repository https://jcoliver.r-universe.dev

RemoteUrl https://github.com/jcoliver/lifer

RemoteRef HEAD

RemoteSha 53ddea32084be1cc1037127a26a69d409fb97bb7

Contents

`RecentNearby` *Recent nearby eBird observations*

Description

Recent nearby eBird observations

Usage

```
RecentNearby(
  key,
  lat = 32.241,
  lng = -110.938,
  dist = 50,
  back = 4,
  hotspot = TRUE,
  include_provisional = FALSE,
  max_tries = 5,
  timeout_sec = 30,
  verbose = TRUE
)
```

Arguments

`key`	Character eBird API key.
`lat`	Numeric latitude; use negative values for southern latitudes (i.e. -46.86, *not* "46.86 S).
`lng`	Numeric longitude; use negative values for western longitudes (i.e. -72.08, *not* "72.08 W").
`dist`	Numeric radius in kilometers of distance from geographic center point given by `lat` and `lng` from which to return recent observations.
`back`	Integer number of days back to search for observations.
`hotspot`	Logical indicating whether or not to restrict results to hotspot locations.
`include_provisional`	Logical indicating whether or not to include observations which have not yet been reviewed.
`max_tries`	Integer maximum number of query attempts to try.
`timeout_sec`	Integer time to allow before query is aborted.
`verbose`	Logical determining whether or not to print messages during queries.

Details

The function uses the eBird API (see `https://documenter.getpostman.com/view/664302/S1ENwy59/`) to query recent sightings. Queries to the eBird API require a user key; more information on obtaining a key can be found at the eBird API documentation.

Value

An object of class "recent_obs" with the following elements:

query_type The type of query performed.

query_parameters List of query parameters passed in request.

obs data frame of observations returned from query; if no observations are returned, `obs` is `NULL`. Columns include:

- **speciesCode** The (usually) six-letter species code, see `https://science.ebird.org/en/use-ebird-data/the-ebird-taxonomy/`
- **comName** Species' common name.
- **sciName** Species' scientific name.
- **locId** eBird identifier of the location.
- **locName** Name of the location.
- **obsDt** Observation date as character string in the format "YYYY-MM-DD HH:MM".
- **howMany** Number of individuals.
- **lat** Decimal latitude.
- **lng** Decimal longitude.
- **obsValid** Logical indicating if observation marked as valid.
- **obsReviewed** Logical indicating if observation has been reviewed.
- **locationPrivate** Logical indicating whether or not location is designated as private.
- **subId** Checklist ID for this observation.

Examples

```
## Not run: 
  # Read eBird key in from file
  key <- scan(file = "ebird-key.txt", what = "character")
  # Search for observations 5 km from lat/lng coordinates
  recent <- RecentNearby(key = key, lat = 32.28, lng = -111.02, dist = 5)

## End(Not run)
```

`RecentNearbySpecies` *Retrieve recent nearby observations of a species*

Description

Retrieve recent nearby observations of a species

Usage

```
RecentNearbySpecies(
  key,
  species_code,
  lat = 32.241,
  lng = -110.938,
  dist = 50,
  back = 4,
  hotspot = TRUE,
  include_provisional = FALSE,
  max_tries = 5,
  timeout_sec = 30,
  verbose = TRUE
)
```

Arguments

`key`	Character eBird API key.
`species_code`	Species code for species of interest; usually a six-character string such as "purmar" or "batpig". See https://science.ebird.org/en/use-ebird-data/the-ebird-taxonomy/ for more information.
`lat`	Numeric decimal degree latitude; use negative values for southern latitudes (i.e. -46.86, *not* "46.86 S").
`lng`	Numeric decimal degree longitude; use negative values for western longitudes (i.e. -72.08, *not* "72.08 W").
`dist`	Numeric radius in kilometers of distance from geographic center point given by `lat` and `lng` from which to return recent observations of a species.
`back`	Integer number of days back to search for observations.
`hotspot`	Logical indicating whether or not to restrict results to hotspot locations.
`include_provisional`	Logical indicating whether or not to include observations which have not yet been reviewed.
`max_tries`	Integer maximum number of query attempts to try.
`timeout_sec`	Integer time to allow before query is aborted.
`verbose`	Logical determining whether or not to print messages during queries.

Details

The function uses the eBird API (see https://documenter.getpostman.com/view/664302/S1ENwy59/) to query recent sightings of a species. Queries to the eBird API require a user key; you can request an eBird API key by logging into your eBird account and navigating to https://ebird.org/api/keygen/. See examples and vignette for using your eBird API key.

Value

An object of class "recent_obs" with the following elements:

query_type The type of query performed.

query_parameters List of query parameters passed in request, including the species code.

obs Data frame of observations returned from query; if no observations are returned, `obs` is `NULL`. Columns include:

- **speciesCode** The (usually) six-letter species code, see `https://science.ebird.org/en/use-ebird-data/the-ebird-taxonomy/`
- **comName** Species' common name.
- **sciName** Species' scientific name.
- **locId** eBird identifier of the location.
- **locName** Name of the location.
- **obsDt** Observation date as character string in the format "YYYY-MM-DD HH:MM".
- **howMany** Number of individuals.
- **lat** Decimal latitude.
- **lng** Decimal longitude.
- **obsValid** Logical indicating if observation marked as valid.
- **obsReviewed** Logical indicating if observation has been reviewed.
- **locationPrivate** Logical indicating whether or not location is designated as private.
- **subId** Checklist ID for this observation.

Examples

```
## Not run:
  # Read eBird key in from file
  key <- scan(file = "ebird-key.txt", what = "character")
  # Search for observations of Verdin within 5 km from lat/lng coordinates
  recent <- RecentNearbySpecies(key = key, species_code = "verdin",
                                lat = 32.28, lng = -111.02, dist = 5)

## End(Not run)
```

`SitesReport` *Create report for sites with most unseen species*

Description

Create report for sites with most unseen species

Usage

```
SitesReport(
  centers,
  ebird_key,
  species_seen,
  center_names = NULL,
  report_filename = "Goals-Report",
  report_dir = getwd(),
  report_format = c("html", "pdf"),
  max_sites = 5,
  dist = 50,
  back = 4,
  hotspot = TRUE,
  include_provisional = FALSE,
  max_tries = 5,
  timeout_sec = 30,
  messages = c("minimal", "none", "verbose"),
  drop_patterns = c("sp.", "/", "Domestic type", "hybrid"),
  include_maps = TRUE
)
```

Arguments

`centers`	Numeric vector or matrix of latitude and longitude coordinates; vector should be of length 2, e.g. `c(latitude, longitude)`, while matrix should have two columns (first column is latitude, second column is longitude).
`ebird_key`	Character vector with eBird API key.
`species_seen`	Character vector of species that have already been seen.
`center_names`	Character vector of names to use for each pair of latitude and longitude coordinates in `centers`.
`report_filename`	Name of output file without file extension (see `report_format`); e.g. if `report_filename` is "sites-2021" and `report_format` is "html", the report will be saved to sites-2021.html.
`report_dir`	Destination folder for the output file; if `NULL`, report will be saved to working directory.
`report_format`	File format for report; takes one of two values: "html" or "pdf".
`max_sites`	Maximum number of sites to return for each pair of coordinates defined in `centers`; maximum is 12.
`dist`	Numeric radius in kilometers of distance from each geographic center point defined by coordinates in `centers` from which to return recent observations.
`back`	Number of days back to search for observations.
`hotspot`	Logical indicating whether or not to restrict results to hotspot locations.
`include_provisional`	Logical indicating whether not to include observations which have not yet been reviewed.

`max_tries`	Maximum number of query attempts to try (only for expert use).
`timeout_sec`	Integer time to allow before query is aborted (only for expert use).
`messages`	Character indicating the degree to which messages are printed during the report assembly process. Options are "minimal", "none", or "verbose".
`drop_patterns`	Character vector of patterns in species' names to exclude certain species from consideration, such as domesticated species, hybrids, and observations not identified to species level (e.g. "Toxostoma sp.").
`include_maps`	Logical vector indicating whether or not to draw maps of identified sites; should be length 1 or the number of centers (i.e. same length as `centers` if `centers` is a vector, same number of rows as `centers` if `centers` is a matrix).

Details

The function uses the eBird API (see `https://documenter.getpostman.com/view/664302/S1ENwy59/`) to build the report. Queries to the eBird API require a user key; you can request an eBird API key by logging into your eBird account and navigating to `https://ebird.org/api/keygen/`. See examples and vignette for using your eBird API key.

Value

Silently returns a list with two named elements:

results_list A list where each element is a list of the results of queries for a center. Each element is a list with two named elements:

- **center_info** A list with latitude (`lat`), longitude (`longitude`), and name `name` of the geographic center.
- **results** A tibble of observations from the top sites (with a maximum number of sites defined by `max_sites`).

report_details A list containing the settings used to build this report, such as days back and distances.

Examples

```
## Not run:
  # Read in data downloaded from eBird
  list_file <- system.file("extdata", "example-list.csv", package = "lifeR")
  user_list <- read.csv(file = list_file)
  # Only common names are required
  my_species <- user_list$Common.Name
  # Read in eBird API key from a text file
  key <- scan(file = "ebird-key.txt", what = "character")

  # A single center requires vector of coordinates
  locs <- c(45, -109)
  SitesReport(centers = locs, ebird_key = key,
  species_seen = my_species)

  # For multiple centers, pass a matrix to centers argument
  loc_mat <- matrix(data = c(33, -109, 39, -119.1), nrow = 2, byrow = TRUE)
  loc_names <- c("Brushy Mountain", "Yerington")
  SitesReport(centers = loc_mat, ebird_key = key,
  species_seen = my_species, center_names = loc_names)

## End(Not run)
```

`SplitNames` *Split vector of names into two-column data frame*

Description

Split vector of names into two-column data frame

Usage

```
SplitNames(x, delim = " - ")
```

Arguments

`x`	Vector of species names, in the format "Common Name - Scientific name".
`delim`	Character separator that delimits common from scientific names.

Details

Names from eBird are returned in a single column as: "Snow Goose - Anser caerulescens". This function provides a means of separating the common name ("Snow Goose") from the scientific name ("Anser caerulescens") into two separate columns.

Value

A data.frame of two columns, `Common` and `Scientific`.

Examples

```
# Read in data downloaded from eBird
user_file <- system.file("extdata", "example-list.csv", package = "lifeR")
user_list <- read.csv(file = user_file)
# Retrieve a two-column data frame with common names and scientific names
species_seen <- SplitNames(x = user_list$Species)
# If only common names are required, refer to \code{Common} column
species_seen <- SplitNames(x = user_list$Species)$Common
```

Index