Package 'rgeospatialquality'

August 29, 2016

Type Package

Title Wrapper for the Geospatial Data Quality REST API

Version 0.3.2

Date 2016-06-27

Description Provides native wrappers for the functions available via the spatial quality REST API. See <http://bit.ly/bioinformatics_btw057> for more information on the API.

License MIT + file LICENSE

LazyData TRUE

Imports httr (>= 1.0.0), jsonlite (>= 0.9.19), plyr (>= 1.8.3)

RoxygenNote 5.0.1

Suggests knitr, roxygen2, rmarkdown, testthat, covr, rgbif (>= 0.9.2), rvertnet, rinat

VignetteBuilder knitr

URL https://github.com/ropenscilabs/rgeospatialquality

BugReports https://github.com/ropenscilabs/rgeospatialquality/issues

NeedsCompilation no

Author Javier Otegui [aut, cre]

Maintainer Javier Otegui <javier.otegui@gmail.com>

Repository CRAN

Date/Publication 2016-06-28 08:28:36

R topics documented:

add_flags *Calculate flags for a set of records and add them to the provided data frame*

Description

add_flags calls the POST method of the API in order to extract the flags for a set of records. **NOTE:** currently, the API imposes a hard-limit of 1000 records per request, to avoid malfunctioning due to some third-party library limtations. This function will not work if a data.frame with more than 1000 rows is provided.

Usage

```
add_flags(indf, guess_fields = FALSE, show_summary = TRUE, quiet = FALSE,
  ...)

## Default S3 method:
add_flags(indf, guess_fields = FALSE, show_summary = TRUE,
  quiet = FALSE, ...)

## S3 method for class 'data.frame'
add_flags(indf, guess_fields = FALSE,
  show_summary = TRUE, quiet = FALSE, ...)
```

Arguments

indf	Required. Properly formatted data frame containing a row per record
guess_fields	Optional. Try or not to guess key fields if names don't follow the DarwinCore standard (see details). Defaults to FALSE, meaning it won't try to guess field names and will throw warnings for each missing field. Set to TRUE to try to guess field names (this will make the function stop if no match can be found for any of the key fields)
show_summary	Optional. Show a summary of the quality flags after the process has finished. Defaults to TRUE
quiet	Optional. Don't show any logging message at all. Defaults to FALSE
...	Any extra parameters for httr POST

Details

Internally, the function takes the provided data.frame, transforms it to JSON and makes a POST request to the underlying API with the JSON object in the body of the request. In order to work properly and give comprehensive results, the data.frame should have the four key fields this API works with. See flags for details. If a field is missing, the function will show a warning. If the name of the fields in your data.frame don't conform to the DarwinCore standard, add_flags can try to map the names in the data.frame to the standard ones if the parameter guess_fields is set

to TRUE. In this case, if there is no match, the function will stop and give instructions on how to resume. If there is, the original name in the data.frame will not change.

After finishing, the function returns the provided data.frame with a new column, flags, which holds for each record a list of the geospatial quality flags. If show_summary is TRUE (default value), it also shows a summary of the results, indicating how many records showed different types of issues.

Value

The provided data frame with the quality flags added as new columns

See Also

format_gq, flags, parse_record

Examples

```
## Not run:
# Using the rgbif package
if (requireNamespace("rgbif", quietly=TRUE)) {
 library("rgbif")

 # Prepare data
 d <- occ_data(scientificName="Apis mellifera", limit=50, minimal=FALSE)
 d <- d$data

 # Format data.frame
 d <- format_gq(d, source="rgbif")

 # Execute the call to the API, showing output and
 # logging information, and store the results
 dd <- add_flags(d)

 # Alternatively, instead of formating with 'format_gq', make the function
 # guess the correct name of the fields.
 dd <- add_flags(d, guess_fields=TRUE)

 # Execute the call without showing summary output, but
 # showing logging information
 dd <- add_flags(d, show_summary=FALSE)

 # Execute the call without showing any logging at all
 # (except errors, obviously)
 dd <- add_flags(d, quiet=TRUE)

 # Data quality output will be stored in a new field called flags
 names(dd$flags)

 # You can check records with certain flags as usual
 # See records with coordinates-country mismatch
 dd[dd$flags$coordinatesInsideCountry == FALSE,]
}
```

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

`flags` *Geospatial Quality Flags*

Description

This document describes how the Geospatial Quality API (service for which this package works as a wrapper) works. Specifically, it describes the data input format and enumerates and briefly describes the possible geospatial quality flags that the API returns.

Input

The API operates on Primary Biodiversity Records, *i.e.*, the most basic, interpretation-free pieces of information about the occurrence of an organism (taxonomic identification, or "what") in a specific place (geospatial location, or "where") and a moment in time (temporal location, or "when"). It feeds on a single record or a set of records that describe such occurrence. Specifically, the geospatial issues service works with the "what" and the "where", meaning it is more useful if such pieces of information are provided to the API.

While there is no minimum set of required values to pass to the service in order for it to work, the amount of tests to be performed depends on the information provided. One can simply call the base URL and it will return a set of empty fields. To fully leverage the potential of the API, however, a user should send a well-defined set of values. These values must conform to the DarwinCore Standard, and currently, the API works on these variables:

decimalLatitude Value for the Latitude in decimal degrees format (e.g. 42.332)

decimalLongitude Value for the Longitude in decimal degrees format (e.g. -1.833)

countryCode 2 character ISO code for the country

scientificName Species the record belongs to

Caveat: while the API accepts scientific names as specified in the DarwinCore Standard, currently some tools only work if the "Genus"+"Specific Epithet" binomial is provided in this field. Thus, instead of "Puma concolor (Linnaeus, 1771)", we recommend using just "Puma concolor" in the 'scientificName' field.

When working with multiple records (using the `add_flags` function in this package), the API accepts more than just these fields. One can send a data.frame with as many columns as he/she wants, and the API will just by-pass them. They will be presented in the output, though, so it is a good practice to include a field with any type of identification (like an occurrenceID).

Output format

The output of the API will always be a JSON document, which will be transformed into a data.frame in this package. Actually, the API will return the same document that was provided, with the addition of the flags element. This new element contains the results of the geospatial and spatio-taxonomic checks the API has performed. The attributes in this flags element are one or more of these (depending on the information provided):

hasCoordinates always. TRUE if coordinates have been sent to the API. FALSE otherwise.

hasCountry always. TRUE if a country value has been sent to the API. FALSE otherwise.

hasScientificName always. TRUE if a scientific name has been sent to the API. FALSE otherwise.

validCoordinates only if hasCoordinates is TRUE. TRUE if supplied values conform to the natural limits of coordinates. FALSE otherwise.

validCountry only if hasCountry is TRUE. TRUE if supplied value corresponds to an existing 2-character code for a country. FALSE otherwise.

highPrecisionCoordinates only if validCoordinates is TRUE. TRUE if coordinates have at least 3 decimal figures. FALSE otherwise.

nonZeroCoordinates only if validCoordinates is TRUE. FALSE if both coordinates are 0. TRUE otherwise.

coordinatesInsideCountry Only if validCoordinates and validCountry are TRUE. TRUE if coordinates fall inside the specified country. FALSE otherwise.

transposedCoordinates Only if coordinatesInsideCountry is FALSE. TRUE if swapping the coordinates makes them right. FALSE otherwise. This operation can be performed along negatedLatitude and negatedLongitude.

negatedLatitude Only if coordinatesInsideCountry is FALSE. TRUE if negating the latitude makes coordinates right. FALSE otherwise. This operation can be performed along transposedCoordinates and negatedLongitude.

negatedLongitude Only if coordinatesInsideCountry is FALSE. TRUE if negating the longitude makes coordinates right. FALSE otherwise. This operation can be performed along transposedCoordinates and negatedLatitude.

distanceToCountryInKm Only if coordinatesInsideCountry, transposedCoordinates, negatedLatitude and negatedLongitude are FALSE. This will show the distance to the closest point of the country boundaries, in Km.

coordinatesInsideRangeMap Only if hasScientificName and validCoordinates are TRUE. TRUE if coordinates fall inside the IUCN range map for the specified species. FALSE otherwise.

distanceToRangeMapInKm Only if coordinatesInsideRangeMap is FALSE. This will show the distance to the closest point of the species range map, in Km.

See Also

add_flags, format_gq, parse_record

format_gq *Prepare data frame for flagging functions*

Description

format_gq renames certain fields to make sure the API knows how to use them. This step is highly recommended for the proper assessment of the provided data.frame.

Usage

```
format_gq(indf, source = NULL, config = NULL, quiet = FALSE, ...)

## Default S3 method:
format_gq(indf, source = NULL, config = NULL,
  quiet = FALSE, ...)

## S3 method for class 'data.frame'
format_gq(indf, source = NULL, config = NULL,
  quiet = FALSE, ...)
```

Arguments

indf	Required. The data.frame on which to operate.
source	Optional. Indicates the package that was used to retrieve the data. Currently accepted values are "rvertnet", "rgbif" or "rinat". Either source, config or individual parameters must be present (see details).
config	Optional. Configuration object indicating mapping of field names from the data.frame to the DarwinCore standard. Useful when importing data multiple times from a source not available via the source argument. Either source, config or individual parameters must be present (see details).
quiet	Optional. Don't show any logging message at all. Defaults to FALSE.
...	Optional. If none of the previous is present, the four key arguments (decimalLatitude, decimalLongitude, countryCode, scientificName) can be put here. See examples.

Details

When invoked, there are three ways of indicating the function how to transform the data.frame: using the source parameter, providing a config object with field mapping, or passing individual values to the mapping function. This is the order in which the function will parse arguments; source overrides config, which overrides other mapping arguments.

source refers to the package that was used to retrieve the data. Currently, three values are supported for this argument: "rgbif", "rvertnet" and "rinat", but many more are on their way.

config asks for a configuration object holding the mapping of the field names. This option is basically a shortcut for those users with custom-formatted data.frames who will use the same mapping

many times, to avoid having to type them each time. In practice, this object is a named list with the following four fields: `decimaLatitude`, `decimaLongitude`, `countryCode` and `scientificName`. Each element must have a string indicating the name of the column in the data.frame holding the values for that element. If the data.frame doesn't have one or more of these fields, put `NA` in that element; otherwise, the function will throw an error. See the examples section.

If none of the two is provided, the function expects the user to provide the mapping by passing the individual column names associated with the right term of the DarwinCore Standard. See the examples section.

Value

The provided data frame, with field names changed to fit the API functioning.

See Also

`add_flags`

Examples

```
## Not run:
# Using the rgbif package and the source argument
if (requireNamespace("rgbif", quietly=TRUE)) {
 d <- rgbif::occ_data(scientificName="Apis mellifera", limit=50, minimal=FALSE)
 d <- d$data
 d <- format_gq(d, source="rgbif")

 # Using a configuration object (matches 'rinat' schema)
 conf <- list(decimalLatitude="latitude",
              decimalLongitude="longitude",
              countryCode=NULL,
              scientificName="scientific_name")
 d <- format_gq(d, config=conf)

 # Passing individual parameters, all optional
 d <- format_gq(d,
                decimalLatitude="lat",
                decimalLongitude="lng",
                countryCode="ccode",
                scientificName="sciname")
}

## End(Not run)
```

`parse_record` *Calculate flags for a single record*

Description

`parse_record` calls the `GET` method of the API in order to extract the flags for an individual record. It returns just the `flags` element.

Usage

```
parse_record(record = NULL, decimalLatitude = NULL,
  decimalLongitude = NULL, countryCode = NULL, scientificName = NULL, ...)
```

Arguments

record	List-type object containing information on the record. If present, it MUST contain the following four attributes as named elements.
decimalLatitude	Only if 'record' is not present. Latitude in decimal degrees format (float, e.g. 42.1833)
decimalLongitude	Only if 'record' is not present. Longitude in decimal degrees format (float, e.g. -1.8332)
countryCode	Only if 'record' is not present. Two or three-letter ISO-3166 country codes (string, e.g. "ES")
scientificName	Only if 'record' is not present. Binomial identifying the species (string, e.g. "Puma concolor")
...	Any extra parameters for `httr` GET

Details

Data can be passed in two different ways to this function: either with the four key elements (`decimalLatitude`, `decimalLongitude`, `countryCode` and `scientificName`) passed as named arguments or via a single `record` parameter that consists of a list with these four elements. However, if both are filled, the function will stop and show an error message. The more filled fields, the more informative the output of the API will be. However, due to the flexible nature of the underlying API, a valid response will be get even if no input data is provided. Therefore, calling `parse_record()` with no arguments will return a list of three `FALSE` elements, which represents no information at all. The function will however throw a warning for each missing field. See `flags` for more info on the input and output of the function.

Value

A named list with the geospatial quality flags

See Also

`flags`, `add_flags`

Examples

```
## Not run:
# Using the 'record' parameter
# Create 'record' object with values
rec <- list(decimalLatitude=42.1833, decimalLongitude=-1.8332,
            countryCode="ES", scientificName="Puma concolor")
# Call the API and get the results
```

```
parse_record(record=rec)

# Using named arguments
parse_record(decimalLatitude=42.1833, decimalLongitude=-1.8332,
           countryCode="ES", scientificName="Puma concolor")
# In both cases, the result will be the same

# If any parameter is missing, the function runs, but throws a warning
# message. This will throw a warning saying that 'countryCode' and
# 'scientificName' fields are missing. Also, the results will be
# limited to the feasible calculations.
parse_record(decimalLatitude=42.1833, decimalLongitude=-1.8332)

# One can call the function without parameters. Although valid,
# this way of calling the function is useless.
parse_record()

## End(Not run)
```

`rgeospatialquality` *rgeospatialquality: A wrapper for the Geospatial Quality REST API for primary biodiversity data*

Description

This package provides R-native access to the methods of the Geospatial Quality API

Information

`flags` - Information on the input and output of the Geospatial Quality API

Functions

`parse_record` Performs the assessment of a single record

`add_flags` Performs the assessment of a data.frame with one or more records

`format_gq` Adapts the supplied data.frame to the needs of the Geospatial Quality API

Vignettes (published in RPubs)

Transcription of the scientific paper describing the Geospatial Quality API

Using `rgeospatialquality` together with `rgbif`

Index