Package 'rangeBuilder'

December 10, 2019

Type Package

Title Occurrence Filtering, Geographic and Taxonomic Standardization and Generation of Species Range Polygons

Version 1.5

Date 2019-12-09

Author Pascal Title

Maintainer Pascal Title <ptitle@umich.edu>

Imports alphahull, stringi, rgeos (>= 0.1-4), pbapply, cleangeo, methods, Rcpp (>= 0.12.9)

Depends sp, raster, rgdal, R (>= 2.10)

Description Provides tools for filtering occurrence records, generating alpha-hull-derived range polygons and mapping species distributions.

License ACM

URL https://github.com/ptitle/rangeBuilder

NeedsCompilation yes

LinkingTo Rcpp

LazyData true

Repository CRAN

Date/Publication 2019-12-10 11:30:05 UTC

R topics documented:

`rangeBuilder-package` *rangeBuilder*

Description

Provides tools for filtering occurrence records, standardizing countries and species names, generating alpha-hull-derived range polygons and mapping species distributions.

Details

Package:	rangeBuilder
Type:	Package
Version:	1.4
Date:	2017-05-31
License:	GPL-2 | GPL-3

Author(s)

Pascal Title <ptitle@umich.edu>

References

Davis Rabosky, A.R., C.L. Cox, D.L. Rabosky, P.O. Title, I.A. Holmes, A. Feldman and J.A. McGuire. 2016. Coral snakes predict the evolution of mimicry across New World snakes. Nature Communications 7:11484.

acceptedAndSynonyms *Accepted names and synonyms*

Description

Several functions are provided here to perform simple queries on accepted vs synonymous species names.

Usage

```
getSynonymsFromAccepted(sp, db)
getAcceptedFromSynonym(sp, db)
getAcceptedNames(db)
```

Arguments

sp	genus and species
db	the database to query, can be squamates, birds, mammals or amphibians.

Details

The workhorse function for matching synonyms to accepted names is synonymMatch. The functions here are more simple, and are intended to be complementary to the main matching function.

The squamate database is a local copy of the Reptile Database (http://reptile-database.reptarium.cz/), which will be updated periodically. The list of accepted names within this R package are those that are listed as such on the website.

The bird database is the BirdLife Taxonomic Checklist v8.0 as downloaded from http://www.birdlife.org/datazone/info/taxonomy.

The mammal database is Wilson and Reeder's Mammal Species of the World, 3rd edition, downloaded from http://www.departments.bucknell.edu/biology/resources/msw3/.

The amphibian database is a local copy of the AmphibiaWeb taxonomy (https://amphibiaweb.org/taxonomy/index.html), which will be updated periodically.

To see when these datasets were last updated for this R package, run downloadDates.

Citation:

BirdLife International. 2015. The BirdLife checklist of the birds of the world: Version 8. Downloaded from http://www.birdlife.org/datazone/userfiles/file/Species/Taxonomy/BirdLife_Checklist_Version_80.zip [.xls zipped 1 MB].

Don E. Wilson & DeeAnn M. Reeder (editors). 2005. Mammal Species of the World. A Taxonomic and Geographic Reference (3rd ed), Johns Hopkins University Press, 2,142 pp.

Uetz P., Hosek, J. (ed.). 2016. The Reptile Database, http://www.reptile-database.org (accessed 30 April 2016).

Value

`getSynonymsFromAccepted` returns a vector of synonyms for the specified accepted species name.

`getAcceptedFromSynonym` returns the accepted names that have the specified species name as a synonym (as per strict matching).

`getAcceptedNames` returns the list of accepted species names in the database.

Author(s)

Pascal Title

See Also

`synonymMatch`

Examples

```
getSynonymsFromAccepted('Phrynosoma_coronatum', db = 'squamates')
getAcceptedFromSynonym('Phrynosoma_jamesi', db = 'squamates')
```

---

addRasterLegend *Add a legend to a raster plot*

Description

Adds a legend to an existing raster plot, with some more intuitive control than the default legend.

Usage

```
addRasterLegend(r, direction, side, location = 'right', nTicks = 2,
shortFrac = 0.02, longFrac = 0.3, axisOffset = 0, border = TRUE,
ramp = "terrain", isInteger = 'auto', ncolors = 64, breaks = NULL,
minmax = NULL, locs = NULL, cex.axis = 0.8, labelDist = 0.7, digits = 2, ...)
```

Arguments

`r`	the rasterLayer object that has been plotted
`direction`	direction of color ramp. If omitted, then direction is automatically inferred, otherwise can be specified as `horizontal` or `vertical`.
`side`	side for tick marks, see `axis` documentation. Automatically inferred if omitted.
`location`	either a location name (see `Details`), or coordinates for the corners of the bar legend `c(xmin,xmax,ymin,ymax)`.
`nTicks`	number of tick marks, besides min and max.
`shortFrac`	Percent of the plot width range that will be used as the short dimention of the legend. Only applies to preset location options.

longFrac	Percent of the plot width range that will be used as the long dimention of the legend. Only applies to preset location options.
axisOffset	distance from color bar for labels, as a percent of the plot width.
border	logical, should the color legend have a black border
ramp	either a vector of color names for defining the color ramp, or "terrain" (default raster behavior)
isInteger	If auto, automatically determines if raster is made up of integer values, otherwise TRUE or FALSE
ncolors	grain size of color ramp
breaks	If a custom set of color breaks were used in plotting the raster, pass those color breaks here. This overrides the minmax option.
minmax	min and max values from which the color ramp will be derived. If left as NULL, the min and max of the raster will be used.
locs	locations of tick marks, if NULL automatically placed
cex.axis	size of axis labels
labelDist	distance from axis to axis labels (passed to mgp)
digits	number of decimal places for labels
...	additional parameters to be passed to axis.

Details

A number of predefined locations exist in this function to make it easy to add a legend to a raster plot. Preset locations are: topleft, topright, bottomleft, bottomright, left, right, top and bottom. If more fine-tuned control is desired, then a numeric vector of length 4 can be supplied to location, specifying the min x, max x, min y and max y values for the legend.

See examples.

Value

Invisibly returns a list with the following components.

coords	2-column matrix of xy coordinates for each color bin in the legend.
width	Coordinates for the short dimension of the legend.
pal	the color ramp
tickLocs	the tick mark locations in plotting units
labels	the values associated with those tick locations.

Author(s)

Pascal Title

Examples

```
r <- raster(system.file("external/test.grd", package="raster"))

plot(r, legend = FALSE)
addRasterLegend(r, location = 'right')
addRasterLegend(r, location = 'top')

#fine-tune placement
plot(r, legend = FALSE)
addRasterLegend(r, location=c(181000, 181100, 330500, 331500), side = 4)
```

`closestCountry` *Return country from point*

Description

Determines which country a given point falls in.

Usage

```
closestCountry(pt, proj = "+proj=longlat +datum=WGS84")
```

Arguments

`pt`	longitude and latitude, as a numeric vector, 2-column table, or SpatialPoints object.
`proj`	the proj4string of the coordinate. If `pt` is a SpatialPoints object, proj is ignored.

Details

Based on a predetermined set of global points, this function finds the country of occurrence. This can be useful for checking the validity of a point by comparing the returned country to the country listed with the occurrence record.

If a point falls close to the boundary between two countries, the names of the nearby countries are returned.

This function will not be of much value if the point falls in the ocean, as it will return the country that is closest, regardless of how far away it is.

Value

If one point is provided, a character vector is returned. If multiple points are provided, a list of character vectors is returned.

Author(s)

Pascal Title

Examples

```
#point near a country border
closestCountry(c(-115.436, 32.657))
```

`coordError` *Coordinate error*

Description

Calculates the potential error in coordinates due to lack of coordinate precision.

Usage

```
coordError(coords, nthreads = 1)
```

Arguments

`coords`	longitude and latitude in decimal degrees, either as a long/lat vector, or as a 2-column table. Can be either as numeric or character format.
`nthreads`	number of threads to use for parallelization of the function. The R package `parallel` must be loaded for `nthreads > 1`.

Details

This function assumes that the true precision of the coordinates is equivalent to the greatest number of decimals in either the longitude or latitude that are not trailing zeroes. In other words:
`(-130.45670,45.53000)` is interpreted as `(-130.4567,45.5300)`
`(-130.20000,45.50000)` is interpreted as `(-130.2,45.5)`

If we use `(-130.45670,45.53000)` as an example, these coordinates are interpreted as `(-130.4567,45.5300)` and the greatest possible error is inferred as two endpoints: `(-130.45670,45.53000)` and `(-130.45679,45.53009)`

The distance between these two is then calculated and returned.

Value

Returns a vector of coordinate error in meters.

Author(s)

Pascal Title

Examples

```
data(crotalus)

xy <- crotalus[1:100, c('decimallongitude','decimallatitude')]

coordError(xy)
```

downloadDates *Return download dates of included datasets*

Description

Returns either the specific date that datasets were downloaded, or returns the dataset version.

Usage

```
downloadDates()
```

Value

For the Global Invasive Species Database, the Reptile Database and AmphibiaWeb, the date of download is returned, as these datasets are updated periodically. For the BirdLife Taxonomic Checklist and the Wilson & Reeder Mammals of the World, the version or edition is returned.

Author(s)

Pascal Title

Examples

```
downloadDates()
```

filterByLand *Filter occurrences based on land vs ocean*

Description

Identifies occurrence records that do not occur on land.

Usage

```
filterByLand(coords, proj = '+proj=longlat +datum=WGS84')
```

Arguments

coords	coordinates in the form of a 2 column numeric matrix, data.frame, numeric vector, or SpatialPoints object. If Spatial object, proj4string must be specified.
proj	proj4string of input coords. Ignored if input coords are spatial object.

Details

This function uses a rasterized version of the GSHHG (global self-consistent, hierarchical, high-resolution geography database, https://www.soest.hawaii.edu/pwessel/gshhg/), that has been buffered by 2 km.

Value

returns a logical vector where TRUE means the point falls on land.

Author(s)

Pascal Title

Examples

```
data(crotalus)

#identify points that fall off land
filterByLand(crotalus[,c('decimallongitude','decimallatitude')])
```

`filterByProximity` *Filter by proximity*

Description

Filter occurrence records by their proximity to each other.

Usage

```
filterByProximity(xy, dist, mapUnits = FALSE, returnIndex = FALSE)
```

Arguments

xy	longitude and latitude in decimal degrees, either as class matrix, SpatialPoints or SpatialPointsDataFrame.
dist	minimum allowed distance
mapUnits	if TRUE, distance is interpreted in map units, distance in kilometers if FALSE
returnIndex	if TRUE, will return indices of points that would be dropped, if FALSE, returns the points that satisfy the distance filter.

Details

This function will discard coordinates that fall within a certain distance from other points.

Value

If returnIndex = TRUE, returns a numeric vector of indices. If returnIndex = FALSE, returns coordinates of the same class as the input.

Author(s)

Pascal Title

Examples

```
data(crotalus)

# within the first 100 points in the dataset, identify the set of points to
# drop in order to have points no closer to each other than 20 km

subset <- crotalus[1:100,]
tooClose <- filterByProximity(xy= subset[ ,c('decimallongitude','decimallatitude')],
dist=20, mapUnits = FALSE, returnIndex = TRUE)

plot(subset[ ,c('decimallongitude','decimallatitude')], pch=1, col='blue', cex=1.5)
points(subset[tooClose, c('decimallongitude','decimallatitude')], pch=20, col='red')
```

---

flipSign *Flip sign of coordinates*

Description

Checks for coordinate sign mistakes by checking all possibilities against country occupancy.

Usage

```
flipSign(coordVec, country, returnMultiple = FALSE, filterByLand = TRUE,
proj = "+proj=longlat +datum=WGS84")
```

Arguments

coordVec	numeric vector of length 2: longitude, latitude
country	the country that is associated with the record
returnMultiple	if multiple sign flips lead to the correct country, return all options. If FALSE, returns the coords with the fewest needed sign flips.
filterByLand	if TRUE, alternative coords will be tested for whether or not they fall on land.
proj	the proj4string of the coordinate.

Details

This function generates all possible coordinates with different signs, and runs closestCountry on each, returning the coordinates that lead to a country match. It ignores coordinate options that do not pass filterByLand.

If a point falls close to the boundary between two countries, it is still considered a match.

Value

list with 2 elements

matched	logical: Was the country matched
newcoords	matrix of coordinates that were successful.

Author(s)

Pascal Title

Examples

```
#correct coordinates
flipSign(c(4.28, 39.98), country = 'Spain')

#mistake in coordinate sign
flipSign(c(115.436, 32.657), country = 'United States')

#incorrect sign on both long and lat, but not possible to distinguish for longitude
#except when we consider which alternative coords fall on land.
flipSign(c(-4.28, -39.98), country = 'Spain', filterByLand = FALSE, returnMultiple = TRUE)
flipSign(c(-4.28, -39.98), country = 'Spain', returnMultiple = TRUE)

#coordinates are incorrect
flipSign(c(4.28, 59.98), country = 'Spain')
```

---

getDynamicAlphaHull *Generate polygon based on alpha hulls*

Description

Generates an apha hull polygon, where the alpha parameter is determined by the spatial distribution of the coordinates.

Usage

```
getDynamicAlphaHull(x, fraction = 0.95, partCount = 3, buff = 10000,
initialAlpha = 3, coordHeaders = c('Longitude', 'Latitude'),
clipToCoast = 'terrestrial', proj = "+proj=longlat +datum=WGS84",
alphaIncrement = 1, verbose = FALSE)
```

Arguments

x	dataframe of coordinates in decimal degrees, with a minimum of 3 rows.
fraction	the minimum fraction of occurrences that must be included in polygon.
partCount	the maximum number of disjunct polygons that are allowed.
buff	buffering distance in meters
initialAlpha	the starting value for alpha
coordHeaders	the column names for the longitude and latitude columns, respectively. If x has two columns, these are assumed to be longitude and latitude, and coordHeaders is ignored.
clipToCoast	Either "no" (no clipping), "terrestrial" (only terrestrial part of range is kept) or "aquatic" (only non-terrestrial part is clipped). See Details.

proj	the projection information for x. The default is currently the only supported option.
alphaIncrement	the amount to increase alpha with each iteration
verbose	prints the alpha value to the console, intended for debugging.

Details

From a set of coordinates, this function will create an alpha hull with alpha = initialAlpha, and will then increase alpha by alphaIncrement until both the fraction and partCount conditions are met.

If the conditions cannot be satisfied, then a minimum convex hull is returned.

If clipToCoast is set to "terrestrial" or "aquatic", the resulting polygon is clipped to the coastline, using the gshhs dataset provided with this package.

Value

a list with 2 elements:

hull	a SpatialPolygons object
alpha	the alpha value that was found to satisfy the criteria. If a convex hull was returned, this will list MCH.

Author(s)

Pascal Title

See Also

Alpha hulls are created with ahull.

Examples

```
data(crotalus)

# create a polygon range for Crotalus atrox
x <- crotalus[which(crotalus$genSp == 'Crotalus_atrox'),]
x <- x[sample(1:nrow(x), 50),]

range <- getDynamicAlphaHull(x, coordHeaders=c('decimallongitude','decimallatitude'),
clipToCoast = 'no')

plot(range[[1]], col=transparentColor('dark green', 0.5), border = NA)
points(x[,c('decimallongitude','decimallatitude')], cex = 0.5, pch = 3)

# to add a basic coastline
# plot(gshhs, add = TRUE)
```

getExtentOfList *Get extent of list of SpatialPolygons*

Description

Returns the extent that encompasses all SpatialPolygons in a list

Usage

```
getExtentOfList(shapes)
```

Arguments

shapes	a list of SpatialPolygons

Value

an object of class extent

Author(s)

Pascal Title

Examples

```
data(crotalus)

# create some polygons, in this case convex hulls
sp <- split(crotalus, crotalus$genSp)
sp <- lapply(sp, function(x) x[,c('decimallongitude','decimallatitude')])
sp <- lapply(sp, function(x) x[chull(x),])
poly <- lapply(sp, function(x)
SpatialPolygons(list(Polygons(list(Polygon(x)), ID = 1))))

getExtentOfList(poly)
```

queryGISD *Query the Global Invasive Species Database*

Description

Returns a list of countries, categorized as native and invasive range.

Usage

```
queryGISD(species)
```

Arguments

species	genus and species

Details

This function returns distribution information as found on the Distribution tab from the Global Invasive Species Database: http://www.issg.org/database/welcome/

Because of how the GISD webservice is designed, it is possible to have the same country listed under both native and invasive distributions. This is because the species in question is native to one part of the country and invasive in another part of that country. See the GISD website for more detailed information.

This function queries a static version of the database, which will be updated periodically.

To see when these datasets were last updated for this R package, run downloadDates.

Value

list with 3 elements

species	the name of the species that was queried.
native	a vector of country names that comprise the native range of the species.
alien	a vector of country names that comprise the alien range of the species.

Author(s)

Pascal Title

Examples

```
# find GISD information for the burmese python
queryGISD('Python_molurus')
```

---

rangeBuilder-data *rangeBuilder datasets*

Description

Included datasets in rangeBuilder

Usage

```
data(crotalus)
data(gshhs)
```

Details

The crotalus dataset is the result of a query for genus Crotalus on the VertNet search portal (http://portal.vertnet.org/search), and has been thinned and lightly filtered, to serve as an example dataset for this package.

The gshhs dataset is a simplified version of the low resolution version of the GSHHG (the Global Self-Consistent, Hierarchical, High-resolution Geography Database) available for download from https://www.soest.hawaii.edu/pwessel/gshhg/.

References

Wessel, P., and W. H. F. Smith, A Global Self-consistent, Hierarchical, High-resolution Shoreline Database, J. Geophys. Res., 101, 8741-8743, 1996.

---

rasterStackFromPolyList

Polygon List to rasterStack

Description

Takes a list of polygons and creates a rasterStack.

Usage

```
rasterStackFromPolyList(polyList, resolution = 50000,
  retainSmallRanges = TRUE, extent = "auto", nthreads = 1)
```

Arguments

polyList	a list of SpatialPolygon objects, named with taxon names
resolution	vertical and horizontal size of raster cell, in units of the polygons' projection
retainSmallRanges	boolean; should small ranged species be dropped or preserved. See details.
extent	if 'auto', then the maximal extent of the polygons will be used. If not auto, must be a numeric vector of length 4 with minLong, maxLong, minLat, maxLat.
nthreads	number of threads to use for parallelization of the function. The R package parallel must be loaded for nthreads > 1.

Details

In the rasterization process, all cells for which the polygon covers the midpoint are considered as present and receive a value of 1. If retainSmallRanges = FALSE, then species whose ranges are so small that no cell registers as present will be dropped. If retainSmallRanges = TRUE, then the cells that the small polygon is found in will be considered as present.

Value

an object of class `RasterStack` where all rasters contain values of either NA or 1.

Author(s)

Pascal Title

Examples

```
## Not run:
data(crotalus)

# standardize species names
crotalus$genSp <- synonymMatch(crotalus$genSp, db='squam')

# get 10 species occurrence sets
uniqueSp <- unique(crotalus$genSp)[1:10]
uniqueSp <- uniqueSp[complete.cases(uniqueSp)]

# create range polygons
ranges <- vector('list', length = length(uniqueSp))
for (i in 1:length(uniqueSp)) {
x <- crotalus[which(crotalus$genSp == uniqueSp[i]),]

ranges[[i]] <- getDynamicAlphaHull(x, coordHeaders = c('decimallongitude',
'decimallatitude'), clipToCoast = 'terrestrial')
}

# name the polygons
names(ranges) <- uniqueSp

# keep only the polygons
ranges <- lapply(ranges, function(x) x[[1]])

# Create a rasterStack with the extent inferred from the polygons, and a cell
# resolution of 0.2 degrees.
# cells with the presence of a species get a value of 1, NA if absent.

rangeStack <- rasterStackFromPolyList(ranges, resolution = 0.2)

# calculate species richness per cell, where cell values are counts of species
richnessRaster <- calc(rangeStack, fun=sum, na.rm = TRUE)

# set values of 0 to NA
richnessRaster[richnessRaster == 0] <- NA

#plot
ramp <- colorRampPalette(c('blue','yellow','red'))
plot(richnessRaster, col=ramp(100))

plot(gshhs, add = TRUE, lwd=0.5)
```

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

`reptileDatabaseCountries` *Squamate distribution data from the Reptile Database*

Description

Get the countries of occurrence for a species, or the species list for a country.

Usage

```
getRepDBcountryList(spname)
getRepDBSpFromCountry(country)
getRepDBcountries()
```

Arguments

`spname`	genus and species
`country`	country name

Details

These functions allow you to query country-level distribution information as per the Reptile Database http://reptile-database.reptarium.cz/.

If you see any errors that conflict with what is found online, please inform the package maintainer.

Value

`getRBcountryList` returns a vector of country names for the specified species. `getRepDBSpFromCountry` returns a vector of species names for the specified country. `getRepDBcountries` returns the list of countries that can be queried.

Author(s)

Pascal Title

Examples

```
# return countries of occurrence for Naja naja
getRepDBcountryList('Naja_naja')

#return species that occur in New Zealand
getRepDBSpFromCountry('New Zealand')

#return the list of countries that have such data on Reptile-Database
getRepDBcountries()
```

`standardizeCountry` *Standardize country name*

Description

Standardizes country names to the list of countries used internally by this package.

Usage

```
standardizeCountry(country, fuzzyDist = 1, nthreads = 1)
```

Arguments

`country`	character vector of country names or ISO codes
`fuzzyDist`	for fuzzy searching, the maximum string distance allowed for a match; if 0, fuzzy searching is disabled.
`nthreads`	number of threads to use for parallelization of the function. The R package `parallel` must be loaded for `nthreads > 1`.

Details

This package interacts with data from the Global Invasive Species Database (GISD), the Reptile Database, as well as global maps that were used to generate the internal dataset used by `closestCountry`. Efforts have been made to make country names consistent across these separate datasets. This function can be used to convert the user's `Country` field to the same standardized set.

Fuzzy matching uses the function `adist`.

Parallelization with `nthreads` becomes more time-efficient only if the input vector is of multiple thousands of country names.

Value

Character vector of the standardized country names. If no match found, `""` is returned.

Author(s)

Pascal Title

Examples

```
standardizeCountry(c("Russian Federation", "USA", "Plurinational State of Bolivia", "Brezil"))
```

synonymMatch *Match synonyms to accepted names*

Description

Performs strict and fuzzy matching to return the accepted species name

Usage

```
synonymMatch(sp, db, fuzzy = TRUE, fuzzyDist = 2, advancedSearch = TRUE,
searchSynonyms = TRUE, year1=1950, year2=1900, returnMultiple = FALSE,
printReport = TRUE, nthreads = 1)
```

Arguments

sp	a character vector of Genus_species (can be multiple)
db	squamates, birds, mammals, amphibians
fuzzy	logical, should fuzzy matching be used
fuzzyDist	for fuzzy searching, the maximum string distance allowed for a match
advancedSearch	logical, should advanced searching be used, see Details.
searchSynonyms	if FALSE, strict and fuzzy matching is applied only to the list of accepted names
year1	specific to squamates, year for oldest considered synonyms, see details
year2	specific to squamates, year for oldest considered synonyms, second pass
returnMultiple	if FALSE, NA is returned if no match found or if multiple matches found. if TRUE, then multiple hits are returned.
printReport	if TRUE, a summary report is printed at the end of the run.
nthreads	number of threads to use for parallelization of the function. The R package parallel must be loaded for nthreads > 1.

Details

The order of the procedure applied here is as follows:
Strict matching against accepted names,
fuzzy matching against accepted names,
strict matching against synonyms from year1 to present,
fuzzy matching against synonyms from year1 to present,
AdvancedSearch:
strict matching against synonyms from year2 to present,
consideration of alternate latin suffixes and all genus/species combinations with strict matching,
consideration of alternate latin suffixes and all genus/species combinations with fuzzy matching.

The printed report shows counts for the set of unique taxon names, not the full vector that was input.

Parallelization becomes time-efficient with as few as 15 unique taxon names.

The squamate database is a local copy of the Reptile Database (http://reptile-database.reptarium.cz/), which will be updated periodically. The list of accepted names within this R package are those that are listed as such on the website.

The bird database is the BirdLife Taxonomic Checklist v8.0 as downloaded from http://www.birdlife.org/datazone/info/taxonomy.

The mammal database is Wilson and Reeder's Mammal Species of the World, 3rd edition, downloaded from http://www.departments.bucknell.edu/biology/resources/msw3/.

The amphibian database is a local copy of the AmphibiaWeb taxonomy (https://amphibiaweb.org/taxonomy/index.html), which will be updated periodically.

To see when these datasets were last updated for this R package, run downloadDates.

Citation:

BirdLife International. 2015. The BirdLife checklist of the birds of the world: Version 8. Downloaded from http://www.birdlife.org/datazone/userfiles/file/Species/Taxonomy/BirdLife_Checklist_Version_80.zip [.xls zipped 1 MB].

Don E. Wilson & DeeAnn M. Reeder (editors). 2005. Mammal Species of the World. A Taxonomic and Geographic Reference (3rd ed), Johns Hopkins University Press, 2,142 pp.

Uetz P., Hosek, J. (ed.). 2016. The Reptile Database, http://www.reptile-database.org.

Value

a vector of matches, NA if the species name could not be unambiguously matched to a single accepted name. If returnMultiple = TRUE, then NA is only returned when the taxon name is not found at all in the database.

Author(s)

Pascal Title

Examples

```
# simple misspelling
synonymMatch('Crotalus_atrix', db = 'squamates')

# synonym
synonymMatch('Pipistrellus_macrotis', db = 'mammals')

#synonym with slight misspelling
synonymMatch('Tangara_pulchirrima', db = 'birds')

#no match, but return multiple
synonymMatch('Masticophis_flagellum', db = 'squamates', returnMultiple = TRUE)
```

`transparentColor` *Define colors with transparency*

Description

Converts a named color and opacity and returns the proper RGB code.

Usage

```
transparentColor(namedColor, alpha = 0.8)
```

Arguments

`namedColor`	a color name
`alpha`	a transparency value between 0 and 1, where 0 is fully transparent

Value

Returns the transparent color in RGB format.

Author(s)

Pascal Title

Index