Package ‘spocc’

February 20, 2015

Title R Interface to Species Occurrence Data Sources

Description A programmatic interface to many species occurrence data sources, including GBIF, USGS’s BISON, iNaturalist, Berkeley Ecoinformatics Engine eBird, AntWeb, and more as sources become easily available. This package includes functionality for retrieving species occurrence data, combining that data, and creating various kinds of static and interactive maps.

Version 0.2.4

License MIT + file LICENSE

Date 2015-01-21

URL https://github.com/ropensci/spocc

BugReports https://github.com/ropensci/spocc/issues

LazyData true

VignetteBuilder knitr

Depends ggplot2

Imports rgbif (>= 0.7.7), rinat (>= 0.1.4), rbison (>= 0.3.2), ecoengine (>= 1.4), rebird (>= 0.1.1), AntWeb (>= 0.6.1), ggmap, maptools, rworldmap, sp, rgeos, lubridate, RColorBrewer, httr, rgdal, leafletR (>= 0.2-0), whisker, gistr

Suggests roxygen2, testthat, knitr, taxize

Author Scott Chamberlain [aut, cre],
    Karthik Ram [aut],
    Ted Hart [aut]

Maintainer Scott Chamberlain <myrmecocystus@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2015-01-22 09:24:23
R topics documented:

- spocc-package .......................................................... 2
- basemaps ................................................................. 3
- bbox2wkt ................................................................. 3
- clean_spocc ............................................................. 4
- fixnames ................................................................. 5
- get.palette ............................................................. 6
- mapggplot .............................................................. 7
- mapgist ................................................................. 8
- mapleaflet .............................................................. 9
- occ ................................................................. 10
- occ2df ................................................................. 15
- occ_names ............................................................ 16
- occ_names_options .................................................. 17
- occ_options .......................................................... 18
- occ_to_sp ............................................................ 19
- palettes .............................................................. 20
- spocc_duplicates ...................................................... 20
- spocc_stylegeojson .................................................. 21
- spocc_togejson ....................................................... 21
- wkt_vis .............................................................. 23

Index 24

---

spocc-package

R interface to many species occurrence data sources

---

Description

You don’t need API keys for any of the data providers thus far. However, some may be added in the future that require authentication, just FYI.

Details

Currently supported species occurrence databases:

<table>
<thead>
<tr>
<th>Provider</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>GBIF</td>
<td><a href="http://www.gbif.org/">http://www.gbif.org/</a></td>
</tr>
<tr>
<td>BISON</td>
<td><a href="http://bison.usgs.orl.gov/">http://bison.usgs.orl.gov/</a></td>
</tr>
<tr>
<td>iNaturalist</td>
<td><a href="http://www.inaturalist.org/">http://www.inaturalist.org/</a></td>
</tr>
<tr>
<td>Berkeley ecoengine</td>
<td><a href="https://ecoengine.berkeley.edu/">https://ecoengine.berkeley.edu/</a></td>
</tr>
<tr>
<td>AntWeb</td>
<td><a href="http://www.antweb.org/">http://www.antweb.org/</a></td>
</tr>
</tbody>
</table>

Author(s)

Scott Chamberlain <myrmecocystus@gmail.com>
Description

Base maps to use with maprcharts function

Usage

basemaps()

bbox2wkt

Convert a bounding box to a Well Known Text polygon, and a WKT to a bounding box

Description

Convert a bounding box to a Well Known Text polygon, and a WKT to a bounding box

Usage

bbox2wkt(minx = NA, miny = NA, maxx = NA, maxy = NA, bbox = NULL)

wkt2bbox(wkt = NULL)

Arguments

minx Minimum x value, or the most western longitude
miny Minimum y value, or the most southern latitude
maxx Maximum x value, or the most eastern longitude
maxy Maximum y value, or the most northern latitude
bbox A vector of length 4, with the elements: minx, miny, maxx, maxy
wkt A Well Known Text object.

Value

bbox2wkt returns an object of class character, a Well Known Text string of the form 'POLYGON((minx miny, maxx miny, maxx maxy, minx maxy, minx miny))'.
wkt2bbox returns a numeric vector of length 4, like c(minx, miny, maxx, maxy).
Examples

```r
## Not run:
# Convert a bounding box to a WKT

## Pass in a vector of length 4 with all values
mm <- bbox2wkt(bbox=c(38.4,-125.0,40.9,-121.8))
plot(e)

## Or pass in each value separately
mm <- bbox2wkt(minx=38.4, miny=-125.0, maxx=40.9, maxy=-121.8)
plot(readWKT(mm))

# Convert a WKT object to a bounding box
wkt <- "POLYGON((38.4 -125,40.9 -125,40.9 -121.8,38.4 -121.8,38.4 -125))"
wkt2bbox(wkt)

## End(Not run)
```

---

**clean_spocc**

Clean spocc data

**Description**

Clean spocc data

**Usage**

```r
clean_spocc(input, country = NULL, country_which = "include",
            shppath = NULL, habitat = NULL, provider_duplicates = FALSE)
```

**Arguments**

- `input` An object of class occdat
- `country` (logical) Attempt to clean based on country. Ignored for now.
- `country_which` (character) One of include, xxx. Ignored for now.
- `shppath` (character) Path to shape file to check against. Ignored for now.
- `habitat` (character) Attempt to clean based on habitat. Ignored for now.
- `provider_duplicates` (logical) Whether to remove duplicates from the same provider in separate sources. Ignored for now.
Details

We’ll continue to add options for cleaning data, but for now, this function:

- Removes impossible values of latitude and longitude
- Removes any NA values of latitude and longitude
- Removes points at 0,0 - these points are likely wrong

Some examples below don’t actually work yet, but will soon.

Value

Returns an object of class occdat+occlean. See attributes of the return object for details on cleaning results.

Examples

```r
## Not run:
res <- occ(query = c('Ursus','Accipiter','Rubus'), from = 'bison', limit=10)
class(res)
res_cleaned <- clean_spocc(input=res)
class(res_cleaned) # now with classes occdat and occclean

#### THESE AREN'T WORKING...
# Country cleaning
res <- occ(query = 'Ursus americanus', from = 'gbif', limit=500,
            gbifopts = list(hasCoordinate=TRUE))
res$gbif
plot(res)

res <- occ(query = 'Ursus americanus', from = 'gbif', limit=1200,
            gbifopts = list(hasCoordinate=TRUE))
plot(res)
res2 <- clean_spocc(res, country = "Mexico")
plot(res2)

# Clean provider duplicates
res <- occ(query = 'Ursus americanus', from = c('gbif','inat'), limit=300)
plot(res)
res2 <- clean_spocc(input=res, provider_duplicates = TRUE)

## End(Not run)
```

fixnames

Change names to be the same for each taxon.

Description

That is, this function attempts to take all the names that are synonyms, for whatever reason (e.g., some names have authorities on them), and collapses them to the same string - making data easier to deal with for making maps, etc.
get_palette

Get palette actual name from longer names

Description

Get palette actual name from longer names

Usage

fixnames(obj, how = "shortest", namevec = NULL)

Arguments

obj  
An object of class occdat

how  
One of a few different methods:

• shortest Takes the shortest name string that is likely to be the prettiest to display name, and replaces all names with that one, better for maps, etc.
• query This method takes the names you originally queried on (in the occdat object), and replaces names in the occurrence data with them.
• supplied If this method, supply a vector of names to replace the names with

namevec  
A vector of names to replace names in the occurrence data.frames with. Only used if how="supplied"

Value

An object of class occdat.

Examples

## Not run:
spp <- c('Danaus plexippus', 'Accipiter striatus', 'Pinus contorta')
dat <- occ(spp, from='gbif', gbifopts=list(hasCoordinate=TRUE), limit=50)
fixnames(dat, how="shortest")$gbif
fixnames(dat, how="query")$gbif
fixnames(dat, how="supplied", namevec = c('abc', 'def', 'ghi'))$gbif

dat <- occ(spp, from='ecoengine', limit=50)
## doesn't changes things
fixnames(dat, how="shortest")$ecoengine$data$Danaus_plexippus
## this is better
fixnames(dat, how="query")$ecoengine$data$Danaus_plexippus
## or this
fixnames(dat, how="supplied",
    namevec = c("Danaus", "Accipiter", "Pinus"))$ecoengine$data$Danaus_plexippus

## End(Not run)
**mapggplot**

**Usage**

```r
get_palette(userselect)
```

**Arguments**

- `userselect` : User input

**Description**

A ggplot2 visualization of species occurrences.

**Usage**

```r
mapggplot(df, zoom = 5, point_color = "#86161f")
```

**Arguments**

- `df` : Input data frame
- `zoom` : Zoom level for map. Adjust depending on how your data look.
- `point_color` : Default color of your points

**Examples**

```r
## Not run:
ecoengine_data <- occ(query = 'Lynx rufus californicus', from = 'ecoengine', limit=100)
mapggplot(ecoengine_data)

gbif_data <- occ(query = 'Accipiter striatus', from = 'gbif', limit=100)
mapggplot(gbif_data)

bison_data <- occ(query = 'Accipiter striatus', from = 'bison', limit=100)
mapggplot(bison_data)

## End(Not run)
```
mapgist

Make an interactive map to view in the browser as a Github gist

**Description**

Make an interactive map to view in the browser as a Github gist

**Usage**

```r
mapgist(data, description = "", file = "gistmap", dir = NULL,
        public = TRUE, browse = TRUE, ...)
```

**Arguments**

- `data` A data.frame, with any number of columns, but with at least the following: name (the taxonomic name), latitude (in dec. deg.), longitude (in dec. deg.)
- `description` Description for the Github gist, or leave to default (=no description)
- `file` File name (without file extension) for your geojson file. Default is 'gistmap'.
- `dir` Directory for storing file and reading it back in to create gist. If none is given, this function gets your working directory and uses that.
- `public` (logical) Whether gist is public (default: TRUE)
- `browse` If TRUE (default) the map opens in your default browser.
- `...` Further arguments passed on to spocc_stylegeojson

**Details**

See `gist_auth` for help on authentication

**Examples**

```r
# Not run:
spp <- c("Danaus plexippus",'Accipiter striatus','Pinus contorta')
dat <- occ(spp, from=c('gbif','ecoengine'), limit=30, gbifopts=list(hasCoordinate=TRUE))
dat <- fixnames(dat, "query")

# Define colors
mapgist(data=dat, color=c('#976AAE','#6B944D','#BD5945'))
mapgist(data=dat$gbif, color=c('#976AAE','#6B944D','#BD5945'))
mapgist(data=dat$ecoengine, color=c('#976AAE','#6B944D','#BD5945'))

# Define colors and marker size
mapgist(data=dat, color=c('#976AAE','#6B944D','#BD5945'), size=c('small','medium','large'))

# Define symbols
mapgist(data=dat, symbol=c('park','zoo','garden'))

# End(Not run)
```
mapleaflet

Make an interactive map to view in the browser

Description

Make an interactive map to view in the browser

Usage

mapleaflet(dat, popup = TRUE, map_provider = "osm", zoom = 3,
  title = "map", size, centerview = c(30, -73.9), dest = ".",
  overwrite = TRUE, incl.data = TRUE)

Arguments

dat
  A data.frame, with any number of columns, but with at least the following: name (the taxonomic name), latitude (in dec. deg.), longitude (in dec. deg.)

popup
  If TRUE (default) popup tooltips are created for each point with metadata for that point.

map_provider
  Base map to use. One or a list of 'osm' (OpenStreetMap standard map), 'tls' (Thunderforest Landscape), 'cm' (CloudMade), 'mqosm' (MapQuest OSM) or 'mqsat' (MapQuest Open Aerial). Default is 'osm'. See leaflet for more information.

zoom
  Map zoom, 0 being most zoomed out, and 18 most zoomed out. See leaflet for more information.

title
  Map title

size
  Height and width (in pixels) of map as a length 2 vector. If missing, a fullscreen (browser window) map is generated.

centerview
  Lat/long position to center map

dest
  Specify a path to save an html file of your map. You can open this in your browser to view it. If left as NULL (the default) the map opens up in your default browser, or if you have a newer version of RStudio open in RStudio Viewer pane.

overwrite
  Default: TRUE. Set to FALSE to prevent overwriting local files

incl.data
  Default: TRUE. Writes geoJSON data into the html file to get around security restrictions in browsers like Google Chrome. Set to FALSE to read from a separate local geoJSON file.

Details

NOTE that with some map_provider options you will have no map layer show up at first. This may be because there is no map at that particular zoom level. Just zoom in or out to see the map.
Examples

```r
## Not run:
spp <- c('danaus plexippus', 'accipiter striatus', 'pinus contorta')
dat <- occ(query = spp, from = 'gbif', gbifopts = list(hasCoordinate = TRUE), limit=50)
mapleaflet(dat, dest = '.')
```

```r
# An example with more species, a different base map, and different color palette
spp <- c('danaus plexippus', 'accipiter striatus', 'pinus contorta', 'puma concolor', 'ursus americanus', 'gymnogyps californianus')
dat <- occ(query = spp, from = 'gbif', gbifopts = list(hasCoordinate = TRUE), limit=50)
mapleaflet(dat, map_provider = 'toner')
```

## End(Not run)

---

**occ**

Search for species occurrence data across many data sources.

### Description

Search on a single species name, or many. And search across a single or many data sources.

### Usage

```r
occ(query = NULL, from = "gbif", limit = 500, geometry = NULL,
    rank = "species", type = "sci", ids = NULL, callopts = list(),
    gbifopts = list(), bisonopts = list(), inatopts = list(),
    ebirdopts = list(), ecoengineopts = list(), antwebopts = list())
```

### Arguments

- **query** (character) One to many names. Either a scientific name or a common name. Specify whether a scientific or common name in the type parameter. Only scientific names supported right now.
- **from** (character) Data source to get data from, any combination of gbif, bison, inat, ebird, and/or ecoengine
- **limit** (numeric) Number of records to return. This is passed across all sources. To specify different limits for each source, use the options for each source (gbifopts, bisonopts, inatopts, ebirdopts, ecoengineopts, and antwebopts). See Details for more. Default: 500 for each source. **BEWARE:** if you have a lot of species to query for (e.g., n = 10), that’s 10 * 500 = 5000, which can take a while to collect. So, when you first query, set the limit to something smallish so that you can get a result quickly, then do more as needed.
- **geometry** (character or numeric) One of a Well Known Text (WKT) object or a vector of length 4 specifying a bounding box. This parameter searches for occurrences inside a box given as a bounding box or polygon described in WKT format. A WKT shape written as 'POLYGON((30.1 10.1, 20 20 40, 40 40, 30.1 10.1))' would be queried as is, i.e. http://bit.ly/HwUSif. See Details for more examples
of WKT objects. The format of a bounding box is [min-longitude, min-latitude, max-longitude, max-latitude].

**rank**
(character) Taxonomic rank. Not used right now.

**type**
(character) Type of search: sci (scientific) or com (common name, vernacular). Not used right now.

**ids**
Taxonomic identifiers. This can be a list of length 1 to many. See examples for usage. Currently, identifiers for only 'gbif' and 'bison' for parameter 'from' supported. If this parameter is used, query parameter can not be used - if it is, a warning is thrown.

**callopts**
Options passed on to GET, e.g., for debugging curl calls, setting timeouts, etc. This parameter is ignored for sources: antweb, inat.

**gbifopts**
(list) List of named options to pass on to occ_search. See also occ_options.

**bisonopts**
(list) List of named options to pass on to bison. See also occ_options.

**inatopts**
(list) List of named options to pass on to get_inat_obs. See also occ_options.

**ebirdopts**
(list) List of named options to pass on to ebirdregion or ebirdgeo. See also occ_options.

**ecoengineopts**
(list) List of named options to pass on to ee_observations. See also occ_options.

**antwebopts**
(list) List of named options to pass on to aw_data. See also occ_options.

**Details**

The occ function is an opinionated wrapper around the rgbif, rbison, rinat, rebird, AntWeb, and ecoengine packages to allow data access from a single access point. We take care of making sure you get useful objects out at the cost of flexibility/options - although you can still set options for each of the packages via the gbifopts, bisonopts, inatopts, ebirdopts, ecoengineopts, and antwebopts parameters.

When searching ecoengine, you can leave the page argument blank to get a single page. Otherwise use page ranges or simply "all" to request all available pages. Note however that this may hang your call if the request is simply too large.

The limit parameter is set to a default of 25. This means that you will get up to 25 results back for each data source you ask for data from. If there are no results for a particular source, you’ll get zero back; if there are 8 results for a particular source, you’ll get 8 back. If there are 26 results for a particular source, you’ll get 25 back. You can always ask for more or less back by setting the limit parameter to any number. If you want to request a different number for each source, pass the appropriate parameter to each data source via the respective options parameter for each data source.

WKT objects are strings of pairs of lat/long coordinates that define a shape. Many classes of shapes are supported, including POLYGON, POINT, and MULTIPOLYGON. Within each defined shape define all vertices of the shape with a coordinate like 30.1 10.1, the first of which is the latitude, the second the longitude.

Examples of valid WKT objects:

- 'POLYGON(((30.1 10.1, 10 20, 20 60, 60 60, 30.1 10.1))'
- 'POINT((30.1 10.1))'
- 'LINestring(3 4,10 50,20 25)'
• 'MULTIPOINT((3.5 5.6),(4.8 10.5))'
• 'MULTILINESTRING((3 4,10 50,20 25),(-5 -8,-10 -8,-15 -4))'
• 'MULTIPOLYGON(((1 1,5 1,5 5,1 5,1 1),(2 2,2 3,3 3,2 2 2)),((6 3,9 2,9 4,6 3)))'
• 'GEOMETRYCOLLECTION(POINT(4 6),LINESTRING(4 6,7 10))'

Only POLYGON objects are currently supported.

Getting WKT polygons or bounding boxes. We will soon introduce a function to help you select a bounding box but for now, you can use a few sites on the web.

• Bounding box - http://boundingbox.klokanetech.com/
• Well known text - http://arthur-e.github.io/Wicket/sandbox-gmaps3.html

BEWARE: In cases where you request data from multiple providers, especially when including GBIF, there could be duplicate records since many providers’ data eventually ends up with GBIF. See spocc_duplicates for more.

Examples

```r
## Not run:
# Single data sources
(res <- occ(query = 'Accipiter striatus', from = 'gbif', limit = 50))
res$gbif
(res <- occ(query = 'Accipiter striatus', from = 'ecoengine', limit = 50))
res$ecoengine
(res <- occ(query = 'Accipiter striatus', from = 'ebird', limit = 50))
res$ebird
(res <- occ(query = 'Danaus plexippus', from = 'inat', limit = 50))
res$inat
(res <- occ(query = 'Bison bison', from = 'bison', limit = 50))
res$bison

# Data from AntWeb
# By species
(by_species <- occ(query = "linepithema humile", from = "antweb"))
# or by genus
(by_genus <- occ(query = "acanthognathus", from = "antweb"))

occ(query = 'Setophaga caerulescens', from = 'ebird', ebirdopts = list(region='US'))
occ(query = 'Spinus tristis', from = 'ebird', ebirdopts =
  list(method = 'ebirdgeo', lat = 42, lng = -76, dist = 50))

# You can pass on limit param to all sources even though its a different param in that source
## ecoengine example
res <- occ(query = 'Accipiter striatus', from = 'ecoengine', ecoengineopts=list(limit = 5))
res$ecoengine
## This is particularly useful when you want to set different limit for each source
res <- occ(query = 'Accipiter striatus', from = c('gbif','ecoengine'),
  gbifopts=list(limit = 10), ecoengineopts=list(limit = 5))
res

# Many data sources
(out <- occ(query = 'Pinus contorta', from=c('gbif','bison')))
```
## Select individual elements

```
occ$gbif
occ$gbif$data
```

## Coerce to combined data.frame, selects minimal set of columns (name, lat, long)

```
occ2df(out)
```

## Pass in limit parameter to all sources. This limits the number of occurrences

```
# returned to 10, in this example, for all sources, in this case gbif and inat.
occ(query='Pinus contorta', from=c('gbif','inat'), limit=10)
```

### Geometry

## Pass in geometry parameter to all sources. This constraints the search to the
## specified polygon for all sources, gbif and bison in this example.

```
# Check out http://arthur-e.github.io/Wicket/sandbox-gmaps3.html to get a WKT string
occ(query='Accipiter', from='gbif',
    geometry='POLYGON((30.1 10.1, 10 20, 20 60, 60 60, 30.1 10.1))')
occ(query='Helianthus annuus', from='bison',
    geometry='POLYGON((-110.66 38.84, -110.86 39.37, -110.20 39.17, -110.20 38.90, -110.63 38.67, -111.06 38.84))')
```

## Or pass in a bounding box, which is automatically converted to WKT (required by GBIF)
## via the bbox2wkt function. The format of a bounding box is
## [min-longitude, min-latitude, max-longitude, max-latitude].

```
occ(query='Accipiter striatus', from='gbif', geometry=c(-125.0, 38.4, -121.8, 40.9))
```

### Bounding box constraint with ecoengine

## Use this website: http://boundingbox.klokantech.com/ to quickly grab a bbox.
## Just set the format on the bottom left to CSV.

```
occ(query='Accipiter striatus', from='ecoengine', limit=10,
    geometry=c(-125.0, 38.4, -121.8, 40.9))
```

## lots of results, can see how many by indexing to meta

```
res <- occ(query='Accipiter striatus', from='gbif',
    geometry='POLYGON((-69.9 49.2, -69.9 29.0, -123.3 29.0, -123.3 49.2, -69.9 49.2))')
res$gbif
```

## You can pass in geometry to each source separately via their opts parameter, at
## least those that support it. Note that if you use rinat, you reverse the order, with
## latitude first, and longitude second, but here it's the reverse for consistency across
## the spocc package

```
bounds <- c(-125.0, 38.4, -121.8, 40.9)
occ(query = 'Danaus plexippus', from="inat", geometry=bounds)
```

## Passing geometry with multiple sources

```
occ(query = 'Danaus plexippus', from=c("inat","gbif","ecoengine"), geometry=bounds)
```

## Using geometry only for the query

### A single bounding box

```
occ(geoemtry = bounds, from = "gbif")
```

### Many bounding boxes

```
occ(geoemtry = list(c(-125.0, 38.4, -121.8, 40.9), c(-115.0, 22.4, -111.8, 30.9)), from = "gbif")
```
# Specify many data sources, another example
beerdopts = list(region = 'US'); gbifopts = list(country = 'US')
out <- occ(query = 'Setophaga caerulescens', from = c('gbif', 'inat', 'bison', 'ebird'),
          gbifopts = gbifopts, beerdopts = beerdopts)
occ2df(out)

# Pass in many species names, combine just data to a single data.frame, and
# first six rows
spnames <- c('Accipiter striatus', 'Setophaga caerulescens', 'Spinus tristis')
out <- occ(query = spnames, from = 'gbif', gbifopts = list(hasCoordinate = TRUE))
df <- occ2df(out)
head(df)

# taxize integration
## You can pass in taxonomic identifiers
library("taxize")
(ids <- get_ids(names=c("Chironomus riparius","Pinus contorta"), db = c('itis','gbif'))) occ(ids = ids[[1]], from='bison') occ(ids = ids, from=c('bison','gbif'), limit=20)

(ids <- get_gbifid("Chironomus riparius")) occ(ids = ids, from='gbif', limit=20)

(ids <- get_tsn('Accipiter striatus')) occ(ids = ids, from='bison', limit=20)

# SpatialPolygons/SpatialPolygonsDataFrame integration
## Single polygon in SpatialPolygons class
one <- Polygon(cbind(c(91,90,90,91), c(30,30,32,30)))
spone = Polygons(list(one), "s1") sppoly = SpatialPolygons(list(spone), as.integer(1)) out <- occ(geometry = sppoly, limit=50)
out$gbif$data

## Two polygons in SpatialPolygons class
one <- Polygon(cbind(c(-121.0,-117.9,-121.0,-121.0), c(39.4, 37.1, 35.1, 39.4)))
two <- Polygon(cbind(c(-123.0,-122.3,-124.5,-123.5,-124.1,-123.0),
                      c(44.8,42.9,41.9,42.6,43.3,44.8)))
spone = Polygons(list(one), "s1") sptwo = Polygons(list(two), "s2") sppoly = SpatialPolygons(list(spone, sptwo), 1:2) out <- occ(geometry = sppoly, limit=50)
out$gbif$data

## Two polygons in SpatialPolygonsDataFrame class
sppoly_df <- SpatialPolygonsDataFrame(sppoly, data.frame(a=c(1,2), b=c("a","b"), c=c(TRUE,FALSE), row.names=row.names(sppoly))) out <- occ(geometry = sppoly_df, limit=50)
occ2df

Combine results from occ calls to a single data.frame

Description

Combine results from occ calls to a single data.frame

Usage

occ2df(obj, what = "data")

Arguments

obj Input from occ
what One of data (default) or all (with metadata)

Examples

## Not run:
spnames <- c("Accipiter striatus", "Setopaha caerulescens", "Carduelis tristis")
out <- occ(query=spnames, from='gbif', gbifopts=list(hasCoordinate=TRUE), limit=10)
occ2df(out)

## End(Not run)
occ_names

Search for species names across many data sources.

Description
Search for species names across many data sources.

Usage
occ_names(query = NULL, from = "gbif", limit = 100, rank = "species",
callopts = list(), gbifopts = list(), bisonopts = list(),
ecoengineopts = list())

Arguments
query (character) One to many names. Either a scientific name or a common name. Specify whether a scientific or common name in the type parameter. Only scientific names supported right now.
from (character) Data source to get data from, any combination of gbif, bison, or ecoengine.
limit (numeric) Number of records to return. This is passed across all sources. To specify different limits for each source, use the options for each source (gbifopts, bisonopts, ecoengineopts). See Details for more. This parameter is ignored for ecoengine.
rank (character) Taxonomic rank. Not used right now.
callopts Options passed on to GET, e.g., for debugging curl calls, setting timeouts, etc.
gbifopts (list) List of named options to pass on to name_lookup. See also occ_names_options.
bisonopts (list) List of named options to pass on to bison_tax. See also occ_names_options.
ecoengineopts (list) List of named options to pass on to ee_search. See also occ_names_options.

Details
Not all 6 data sources available from the occ function are available here, as not all of those sources have functionality to search for names.

Examples
## Not run:
# Single data sources
## gbif
(res <- occ_names(query = 'Accipiter striatus', from = 'gbif'))
head(res$gbif$data[[1]])

## bison
(res <- occ_names(query = '*bear', from = 'bison'))
res$bison$data
**Description**

Look up options for parameters passed to each source for `occ_names` function

**Usage**

```r
occ_names_options(from = "gbif", where = "console")
```

**Arguments**

- `from` (character) Data source to get data from, any combination of gbif, bison, and/or ecoengine. Case doesn’t matter.
- `where` (character) One of console (print to console) or html (opens help page, if in non-interactive R session, prints help to console).

**Details**

Any of the parameters passed to e.g. `name_lookup` from the `rgbif` package can be passed in the associated gbifopts list in `occ`.

Note that the from parameter is lowercased within the function and is called through `match.arg` first, so you can match on unique partial strings too (e.g., ‘e’ for ‘ecoengine’).

**Value**

Opens up the documentation for the function that is used internally within the `occ` function for each source.

**Examples**

```r
## Not run:
# opens up documentation for this function
occ_names_options()

# Open up documentation for the appropriate search function for each source
occ_names_options('gbif')
occ_names_options('ecoengine')
occ_names_options('bison')
```
occ_options

Look up options for parameters passed to each source

Description

Look up options for parameters passed to each source

Usage

occ_options(from = "gbif", where = "console")

Arguments

from (character) Data source to get data from, any combination of gbif, bison, inat, ebird, AntWeb, and/or ecoengine. Case doesn’t matter.

where (character) One of console (print to console) or html (opens help page, if in non-interactive R session, prints help to console).

Details

Any of the parameters passed to e.g. occ_search from the rgbif package can be passed in the associated gbiopts list in occ.

Note that the from parameter is lowercased within the function and is called through match.arg first, so you can match on unique partial strings too (e.g., 'e' for 'ecoengine').

Value

Opens up the documentation for the function that is used internally within the occ function for each source.

Examples

## Not run:
# opens up documentation for this function
occ_options()

# Open up documentation for the appropriate search function for each source
occ_options('gbif')
occ_options('ecoengine')
occ_options('AntWeb')
occ_options('antweb')
occ_options('ebird')
Create a spatial points dataframe from a spocc search.

Description
Create a spatial points dataframe from a spocc search.

Usage
occ_to_sp(occObj, coord_string = "+proj=longlat +datum=WGS84", just_coords = FALSE)

Arguments
occObj The results of a spocc search called by occ()
coord_string A valid EPGS coordinate string from the sp package, the default is WGS 84
just_coords Return data frame with species names and provenance or just a spatial points object, which is the default.

Details
This function will return either a spatial points dataframe or spatial points object. Conversion to spatial points objects allows spocc searches to interact with other spatial data sources. More coordinate system codes can be found at the EPGS registry: http://www.epsg-registry.org/

Examples
## Not run:
## See points on a map
library("maptools")
data(wrld_simpl)
plot(wrld_simpl[wrld_simpl$NAME == "United States", ], xlim=c(-70,-60))
out <- occ(query = "Accipiter striatus", from = c("inat","gbif"))
sp_points <- occ_to_sp(out, just_coords=TRUE)
points(sp_points,col=2)
## End(Not run)
**Description**

Palettes to use with maprcharts function

**Usage**

palettes()

---

**spocc_duplicates**  
*A note about duplicate occurrence records*

---

**Description**

BEWARE: spocc provides you a nice interface to many data providers for species occurrence data. However, in cases where you request data from GBIF in addition to other data sources, there could be duplicate records. This is because GBIF is, to use an ecology analogy, a top predator, and pulls in data from lower nodes in the food chain. For example, iNaturalist provides data to GBIF, so if you search for occurrence records for *Pinus contorta* from iNaturalist and GBIF, you could get, for example, 20 of the same records.

We are working on a way to programmatically flag and/or remove these duplicate records. As you could imagine, this is rather difficult as data is often lost in translation, significant digits could change from provider to provider for the same data, etc.

Still, we think a single R interface to many occurrence record providers will provide a consistent way to work with occurrence data, making analyses and visualizations more repeatable across providers.

We are working on a set of tools for cleaning data, as well as removing duplicates in the spocc_clean function - so keep an eye on that.

Do get in touch with us if you have concerns, have ideas for eliminating duplicates, etc, at <support@ropensci.org>, or at the issue tracker for the spocc package https://github.com/ropensci/spocc/issues/new
spocc_stylegeojson  Style a data.frame prior to converting to geojson.

Description
Style a data.frame prior to converting to geojson.

Usage
spocc_stylegeojson(input, var = NULL, var_col = NULL, var_sym = NULL, var_size = NULL, color = NULL, symbol = NULL, size = NULL)

Arguments
- **input**: A data.frame
- **var**: A single variable to map colors, symbols, and/or sizes to.
- **var_col**: The variable to map colors to.
- **var_sym**: The variable to map symbols to.
- **var_size**: The variable to map size to.
- **color**: Valid RGB hex color
- **symbol**: An icon ID from the Maki project [http://www.mapbox.com/maki/](http://www.mapbox.com/maki/) or a single alphanumeric character (a-z or 0-9).
- **size**: One of ‘small’, ‘medium’, or ‘large’

See Also
spocc_togeojson

spocc_togeojson  Convert spatial data files to GeoJSON from various formats.

Description
The web option uses the Ogre web API. Ogre currently has an output size limit of 15MB. See here [http://ogre.adc4gis.com/](http://ogre.adc4gis.com/) for info on the Ogre web API. The local option uses the function `writeOGR` from the package rgdal.

Note that for Shapefiles, GML, MapInfo, and VRT, you need to send zip files to Ogre. For other file types (.bna, .csv, .dgn, .dxf, .gxt, .txt, .json, .geojson, .rss, .georss, .xml, .gmt, .km, .kml, .kmz) you send the actual file with that file extension.

If you’re having trouble rendering geoJSON files, ensure you have a valid geoJSON file by running it through a geoJSON linter [http://geojsonlint.com/](http://geojsonlint.com/).
Usage

```r
spocc_togeojson(input, method = "web", destpath = "~/",
outfilename = "myfile")
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input</td>
<td>The file being uploaded, path to the file on your machine.</td>
</tr>
<tr>
<td>method</td>
<td>One of web or local. Matches on partial strings.</td>
</tr>
<tr>
<td>destpath</td>
<td>Destination for output geojson file. Defaults to your root directory (&quot;~/&quot;).</td>
</tr>
<tr>
<td>outfilename</td>
<td>The output file name, without file extension.</td>
</tr>
</tbody>
</table>

Details

You can use a web interface called Ogre, or do conversions locally using the rgdal package.

See Also

`spocc_stylegeojson`

Examples

```r
# Not run:
file <- '/Users/scottmac2/Downloads/taxon-placemarks-2441176.kml'

# KML type file - using the web method
spocc_togeojson(file, method='web', outfilename='kml_web')

# KML type file - using the local method
spocc_togeojson(file, method='local', outfilename='kml_local')

# Shp type file - using the web method - input is a zipped shp bundle
file <- '~/github/sac/bison.zip'
spocc_togeojson(file, method='web', outfilename='shp_web')

# Shp type file - using the local method - input is the actual .shp file
file <- '~/github/sac/bison-Bison_bison-20130704-120856.shp'
spocc_togeojson(file, method='local', outfilename='shp_local')

# Get data and save map data
splist <- c('Accipiter erythronemius', 'Junco hyemalis', 'Aix sponsa')
keys <- sapply(splist, function(x) gbif_lookup(name=x, kingdom='plants'))$speciesKey,
      USE_NAMES=FALSE)
out <- occ_search(keys, hasCoordinate=TRUE, limit=50, return='data')
dat <- ldply(out)
datgeojson <- spocc_stylegeojson(input=dat, var='name',
                                  color=c('#976AAE','#6B944D','#BD5945'), size=c('small','medium','large'))

# Put into a github repo to view on the web
write.csv(datgeojson, '~/github/sac/mygeojson/rgbif_data.csv')
file <- '~/github/sac/mygeojson/rgbif_data.csv'
```
spocc_togeojson(file, method='web', destpath='~/github/sac/mygeojson/',
    outfilename='rgbif_data')

# Using rCharts' function spocc_create_gist
write.csv(datgeojson, '~/my.csv')

spocc_togeojson(input=file, method='web', outfilename='my')
spocc_create_gist('~/my.geojson', description = 'Map of three bird species occurrences')

## End(Not run)

---

**wkt_vis**

*Visualize well-known text area's on a map.*

**Description**

This can be helpful in visualizing the area in which you are searching for occurrences with the **occ** function.

**Usage**

```r
wkt_vis(x, zoom = 6, maptype = "terrain", which = "interactive")
```

**Arguments**

- **x**
  - Input well-known text area (character)
- **zoom**
  - Zoom level, defaults to 6 (numeric)
- **maptype**
  - Map type, default is terrain (character)
- **which**
  - One of interactive (default) or static. Interactive open Mapbox map in your browser, and static uses ggplot based ggmap package.

**Examples**

```r
# Not run:
poly <- 'POLYGON((-111.06 38.84, -110.80 39.37, -110.20 39.17, -110.20 38.90,
                      -110.63 38.67, -111.06 38.84))'
wkt_vis(poly)
wkt_vis(poly, which='static')

data <- data.frame(tower='poly2',
                    cbind(lon, lat = c(-125 38.4,-125 40.9,-121.8 40.9,-121.8 38.4,-125 38.4)))
wkt_vis(data)

## End(Not run)
```
Index

*Topic** package
  spocc-package, 2

aw_data, 11
basemaps, 3
bbox2wkt, 3
bison, 11
bison_tax, 16

clean_spocc, 4

ebirdgeo, 11
ebirdregion, 11
ee_observations, 11
ee_search, 16

fixnames, 5

GET, 11, 16
get_inat_obs, 11
get_palette, 6
gist_auth, 8

leaflet, 9

mapggplot, 7
mappist, 8
mapleaflet, 9

name_lookup, 16, 17

occ, 10, 16–18, 23
occ2df, 15
occ_names, 16
occ_names_options, 16, 17
occ_options, 11, 18
occ_search, 11, 18
occ_to_sp, 19

palettes, 20

spocc (spocc-package), 2
spocc-package, 2
spocc_duplicates, 12, 20
spocc_stylegeojson, 21
spocc_togeojson, 21, 21
wkt2bbox (bbox2wkt), 3
wkt_vis, 23
writeOGR, 21