Package 'gbif.range'

August 7, 2026

Version 1.9.1

Title Species Range Mapping from GBIF Using Ecoregion Constraints

Depends R (>= 4.0.0)

Imports terra, rgbif, CoordinateCleaner, sf, ClusterR, FNN, geometry, cluster, mclust, methods, utils, zip, class, NMOF, stats, tools, graphics, grDevices

Suggests data.table, knitr, rmarkdown, testthat (>= 3.0.0)

Description Provides a workflow to retrieve, filter, and analyze 'GBIF' occurrence records and to generate ecologically informed species range maps using downloaded or custom ecoregions. The package includes tools for querying the 'GBIF' backbone taxonomy with get_status(), counting or downloading occurrences with get_gbif_count() and get_gbif(), creating custom ecoregion layers with make_ecoreg(), building range maps with get_range(), and evaluating them against independent validation data with evaluate_range() and cv_range(). A disk-based batch workflow (split_gbif_by_species(), species_csvs_to_ranges(), read_range_rds()) supports large multi-species 'GBIF' exports without loading the full table into memory. Bundled ecoregion layers cover terrestrial (Olson et al. 2001 <doi:10.1641/0006-3568(2001)051[0933:TEOTWA]2.0.CO;2>), marine (Spalding et al. 2007 <doi:10.1641/B570707>), and freshwater (Abell et al. 2008 <doi:10.1641/B580507>) realms. The 'GBIF' API is accessed via the 'rgbif' package, and coordinate cleaning uses 'CoordinateCleaner' (Zizka et al. 2019 <doi:10.1111/2041-210X.13152>). The 'GBIF' API is described at <https://www.gbif.org/developer/summary>.

License GPL (>= 3)

URL https://github.com/8Ginette8/gbif.range, https://8ginette8.github.io/gbif.range/

BugReports https://github.com/8Ginette8/gbif.range/issues

Encoding UTF-8

Language en-US

LazyData true

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'make_tiles.R' 'get_gbif.R' 'get_status.R' 'get_doi.R' 'obs_filter.R' 'get_range.R' 'conv_function.R' 'get_gbif_count.R' 'make_ecoreg.R' 'get_ecoreg.R' 'evaluate_range.R' 'data.R' 'make_blocks.R' 'optme.R' 'cv_range.R' 'classes.R' 'helpers.R' 'gbif_file_workflows.R' 'gbif.range-package.R' 'merge_range.R'

RoxygenNote 8.0.0

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Yohann Chauvier [cre, aut] (ORCID: <https://orcid.org/0000-0001-9399-3192>), Oskar Hagen [aut] (ORCID: <https://orcid.org/0000-0002-7931-6571>), Stefan Pinkert [aut] (ORCID: <https://orcid.org/0000-0002-8348-2337>), Camille Albouy [aut] (ORCID: <https://orcid.org/0000-0003-1629-2389>), Fabian Fopp [aut] (ORCID: <https://orcid.org/0000-0003-0648-8484>), Philipp Brun [aut] (ORCID: <https://orcid.org/0000-0002-2750-9793>), Patrice Descombes [aut] (ORCID: <https://orcid.org/0000-0002-3760-9907>), Florian Altermatt [aut] (ORCID: <https://orcid.org/0000-0002-4831-6958>), Loïc Pellissier [aut] (ORCID: <https://orcid.org/0000-0002-2289-8259>), Katalin Csillery [aut] (ORCID: <https://orcid.org/0000-0003-0039-9296>)

Maintainer Yohann Chauvier <yohann.chauvier@wsl.ch>

Repository CRAN

Date/Publication 2026-08-07 20:50:02 UTC

Contents

`gbif.range-package` *gbif.range: Tools for GBIF Retrieval and Ecoregion-Based Species Range Mapping*

Description

`gbif.range` provides a workflow to retrieve occurrence records from GBIF, clean and filter them for spatial analyses, assemble or download ecoregion layers, generate ecologically informed species range maps, and evaluate the resulting maps against independent data.

Details

The package is designed around a typical workflow: (1) inspect or download GBIF records with `get_gbif_count()` and `get_gbif()`, (2) retrieve taxonomic information with `get_status()`, (3) load packaged ecoregions with `read_ecoreg()` or create custom ones with `make_ecoreg()`, (4) build range maps with `get_range()`, and (5) evaluate those maps with `evaluate_range()` or `cv_range()`.

Additional helpers include `get_doi()` for creating GBIF-derived dataset DOIs, `obs_filter()` for grid-based thinning, `make_tiles()` for splitting study extents into GBIF-ready polygons, and a disk-based workflow built around `split_gbif_by_species()`, `species_csvs_to_ranges()`, and `read_range_rds()` for very large downloaded GBIF tables.

Author(s)

Maintainer: Yohann Chauvier <yohann.chauvier@wsl.ch> (ORCID)

Authors:

- Yohann Chauvier <yohann.chauvier@wsl.ch> (ORCID)
- Oskar Hagen <oskar@hagen.bio> (ORCID)
- Stefan Pinkert (ORCID)
- Camille Albouy (ORCID)
- Fabian Fopp (ORCID)

- Philipp Brun (ORCID)
- Patrice Descombes (ORCID)
- Florian Altermatt (ORCID)
- Loïc Pellissier (ORCID)
- Katalin Csillery (ORCID)

See Also

Useful links:

- https://github.com/8Ginette8/gbif.range
- https://8ginette8.github.io/gbif.range/
- Report bugs at https://github.com/8Ginette8/gbif.range/issues

Examples

```
# ------------------------------------------------------------------------
# 1. Minimal in-memory workflow with custom ecoregions
# ------------------------------------------------------------------------

# Create two simple environmental layers on a small synthetic study area.
r1 <- terra::rast(
  ncols = 20, nrows = 20, xmin = 0, xmax = 10, ymin = 0, ymax = 10
)
terra::values(r1) <- rep(seq(0, 1, length.out = 20), each = 20)
r2 <- terra::rast(r1)
terra::values(r2) <- rep(seq(0, 1, length.out = 20), times = 20)
env <- c(r1, r2)

# Derive custom ecoregions and define a small occurrence table in memory.
eco <- make_ecoreg(env = env, nclass = 4)
occ <- data.frame(
  decimalLongitude = c(0.5, 1.2, 2.4, 3.6, 2.8, 6.0, 7.2, 7.4, 7.6, 8.8),
  decimalLatitude = c(1.0, 0.1, 2.3, 2.5, 2.7, 5.0, 7.1, 7.3, 6.5, 7.7)
)

# Build the range directly from the occurrence table and ecoregions.
range_obj <- get_range(
  occ_coord = occ,
  ecoreg = eco,
  verbose = FALSE
)

# Plot the predicted range and overlay the occurrence points.
terra::plot(range_obj$rangeOutput, col = 3, main = "Range Map")
graphics::points(occ, pch = 4)


# ------------------------------------------------------------------------
# 2. Typical online workflow with recent GBIF data
```

```
# ------------------------------------------------------------------------

# Download all GBIF occurrences for one species
obs <- get_gbif("Ailuropoda melanoleuca")

# Load a packaged terrestrial ecoregion layer
eco_terra <- read_ecoreg("eco_terra", save_dir = tempdir())

# Both calls above depend on remote services and return NULL or an empty
# table if those are unavailable, so guard the rest of this section
if (gbif_have(obs, eco_terra)) {

# Remove observation without data, i.e.,
# probably outdated for a threatned species such as the panda
obs <- obs[!is.na(obs$year), ]

# Inspect the GBIF backbone interpretation used by the package.
status <- get_status("Ailuropoda melanoleuca")
status

# Build the range.
panda_range <- get_range(
  occ_coord = obs,
  ecoreg = eco_terra,
  ecoreg_name = "ECO_NAME"
)

# Plot the predicted terrestrial range and the GBIF occurrences.
terra::plot(
  merge_range(panda_range),
  col = 3,
  main = paste("Range:", obs$scientificName[1])
)
graphics::points(
  obs$decimalLongitude,
  obs$decimalLatitude,
  pch = 4,
  col = grDevices::rgb(1, 0, 1, 0.2)
)

}

# ------------------------------------------------------------------------
# 3. Large downloaded GBIF table already stored on disk
# ------------------------------------------------------------------------

if (requireNamespace("data.table", quietly = TRUE)) {
  # Use the bundled GBIF-style example file as a stand-in for a large download.
  gbif_file <- system.file("extdata", "occ_example_2sps.csv", package = "gbif.range")

  # Keep each stage in its own temporary folder so outputs are easy to inspect.
  split_dir <- file.path(tempdir(), "gbif_pkg_split")
  occ_dir <- file.path(tempdir(), "gbif_pkg_occ_min")
```

```
range_dir <- file.path(tempdir(), "gbif_pkg_ranges")

# Remove earlier temporary outputs so rerunning the example starts cleanly.
unlink(split_dir, recursive = TRUE)
unlink(occ_dir, recursive = TRUE)
unlink(range_dir, recursive = TRUE)

# Split the input table into one occurrence file per speciesKey without
# loading the full file into memory.
split_summary <- split_gbif_by_species(
  input_file = gbif_file,
  outdir = split_dir,
  chunk_size = 10,
  sep_in = "\t",
  sep_out = "\t",
  overwrite = TRUE,
  verbose = FALSE
)

split_summary[, c("species_name", "n_records", "species_file")]

# The batch step resolves "eco_terra" with read_ecoreg(), so it needs the
# remote source to be reachable.
if (gbif_have(read_ecoreg("eco_terra", save_dir = tempdir()))) {

range_summary <- species_csvs_to_ranges(
  species_dir = split_dir,
  ecoreg = "eco_terra",
  ecoreg_name = "ECO_NAME",
  outdir = range_dir,
  occ_outdir = occ_dir,
  occ_save_as = "tsv",
  range_save_as = "rds",
  sep_in = "\t",
  overwrite = TRUE,
  degrees_outlier = 30,
  clust_pts_outlier = 2,
  buff_width_point = 1,
  buff_incrmt_pts_line = 0.1,
  buff_width_polygon = 1,
  format = "SpatVector",
  verbose = FALSE
)

range_summary[, c("species_name", "n_points", "range_file")]

# Read the saved ranges back from disk and plot both species together.
range_one <- read_range_rds(range_summary$range_file[1])
range_two <- read_range_rds(range_summary$range_file[2])

combined_ext <- terra::ext(
  min(terra::xmin(range_one$rangeOutput), terra::xmin(range_two$rangeOutput)),
  max(terra::xmax(range_one$rangeOutput), terra::xmax(range_two$rangeOutput)),
```

```
    min(terra::ymin(range_one$rangeOutput), terra::ymin(range_two$rangeOutput)),
    max(terra::ymax(range_one$rangeOutput), terra::ymax(range_two$rangeOutput))
  )

  terra::plot(combined_ext, col = NA, legend = FALSE)
  terra::plot(range_one$rangeOutput, col = grDevices::rgb(0.1, 0.6, 0.2, 0.5), add = TRUE)
  terra::plot(range_two$rangeOutput, col = grDevices::rgb(0.8, 0.3, 0.1, 0.5), add = TRUE)

  }
}
```

---

area_data *Range Area Comparison Data*

Description

A dataset containing range-area estimates for species, comparing `gbif.range`-derived polygons with corresponding IUCN ranges.

Format

A data frame with 3 columns:

species Character string with the scientific name of the species.

gbif.range.km2 Numeric area in square kilometers calculated from `gbif.range` polygons.

iucn.range.km2 Numeric area in square kilometers calculated from IUCN range data.

Source

Calculated with the `gbif.range` package.

---

check_and_get_ecoreg *Check for a Local Ecoregion Layer and Download It if Needed*

Description

Check whether a target ecoregion directory exists and contains at least one shapefile. If not, download the dataset with `get_ecoreg()`.

Usage

```
check_and_get_ecoreg(ecoreg_name = "eco_terra", save_dir = NULL)
```

Arguments

ecoreg_name	Character. File name of the ecoregion dataset to check. See ecoreg_list for valid values.
save_dir	Character. Directory where downloaded files should be stored. Defaults to tempdir() unless the gbif.range.save_dir option or GBIF_RANGE_SAVE_DIR environment variable is set, in which case that location is used instead.

Value

NULL. The function downloads data only when required.

See Also

get_ecoreg() to force a download, and read_ecoreg() which calls this function internally before loading the layer.

Examples

```
check_and_get_ecoreg("eco_terra", save_dir = tempdir())
```

---

cscl *Draw a Custom Legend Bar*

Description

Internal plotting helper adapted from Philipp Brun.

Usage

```
cscl(
  colors,
  crds,
  horiz = FALSE,
  zrng = c(0, 100),
  at = 10 * 0:10,
  labs = NA,
  tickle = 0.2,
  title = 1,
  lablag = 1,
  titlag = 2,
  box = TRUE,
  breaks,
  cx = 0.8,
  tria = "n"
)
```

Arguments

colors	Vector of fill colors.
crds	Numeric vector of length four giving the plotting coordinates of the legend box.
horiz	Logical. Should the legend be drawn horizontally?
zrng	Numeric vector of length two giving the value range represented by the legend.
at	Numeric values at which tick marks should be drawn.
labs	Optional labels for at. If NA, at is used.
tickle	Numeric tick-mark length scaling factor.
title	Legend title.
lablag	Numeric offset multiplier for tick labels.
titlag	Numeric offset multiplier for the title.
box	Logical. Should a border be drawn around the legend?
breaks	Optional custom break points. Must have length length(colors) + 1 if supplied.
cx	Numeric text expansion factor.
tria	Character flag controlling triangular tips at the legend ends.

Value

NULL, invisibly. Called for its side effect of drawing the legend bar on the current plot device.

---

cv_range *Evaluate a Range Map by Cross-Validation*

Description

Rebuild a get_range() model repeatedly from subsets of the occurrence data stored in a getRange object and evaluate each rebuild against held-out observations.

Usage

```
cv_range(
  range_object = NULL,
  cv = "random-cv",
  nfolds = 5,
  nblocks = 2,
  backpoints = 10000,
  verbose = TRUE
)
```

Arguments

range_object	getRange object. Typically returned by get_range().
cv	Character. String specifying the cross-validation strategy: "random-cv" or "block-cv".
nfolds	Numeric. Number of folds.
nblocks	Numeric. Multiplier used when cv = "block-cv" to define the total number of spatial blocks as nfolds * nblocks.
backpoints	Numeric. Number of regularly spaced background points used as pseudo-absences. Default is 10000.
verbose	Logical. Should fold progress messages be printed?

Details

The function rebuilds the range map nfolds times. In each iteration, one fold is reserved for evaluation and the remaining folds are used for training.

Two strategies are available: random cross-validation and spatial block cross-validation. The latter reduces the influence of spatial autocorrelation by grouping nearby observations before splitting them across folds.

Because true absences are generally unavailable, the evaluation uses a regular grid of background points as pseudo-absences and reports precision, sensitivity, specificity, and TSS.

Value

A data frame with one row per fold plus a Mean row, and the columns TP, FA, TA, FP, Precision, Sensitivity, Specificity, and TSS.

References

Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera- Arroita, G., ... & Dormann, C. F. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography, 40(8), 913-929. doi:10.1111/ecog.02881

Chauvier, Y., Zimmermann, N. E., Poggiato, G., Bystrova, D., Brun, P., & Thuiller, W. (2021). Novel methods to correct for observer and sampling bias in presence-only species distribution models. Global Ecology and Biogeography, 30(11), 2312-2325. doi:10.1111/geb.13383

See Also

get_range() to build the range map being evaluated, and make_blocks() for the underlying fold-assignment logic.

Examples

```
# Load available ecoregions
eco_terra <- read_ecoreg(
    ecoreg_name = "eco_terra",
    save_dir = tempdir(),
    format = "sf"
```

```
)

# First download the worldwide observations of the panda from GBIF
obs_am <- get_gbif(sp_name = "Ailuropoda melanoleuca")

# Both calls above depend on remote services and return NULL or an empty
# table if those are unavailable, so guard the rest of the example
if (gbif_have(eco_terra, obs_am)) {

# Build a range map from occurrence points
range_panda <- get_range(
    occ_coord = obs_am,
    ecoreg = eco_terra,
    ecoreg_name = "ECO_NAME",
    format = "sf"
)
am_test <- cv_range(
    range_object = range_panda,
    cv = "block-cv",
    nfolds = 5,
    nblocks = 2
);am_test

}
```

`ecoreg_list` *Metadata for Downloadable Ecoregion Layers*

Description

A named list describing the ecoregion layers that can be downloaded with `get_ecoreg()`. Each element is itself a list with three character fields: `filename` (the local save name), `link` (the download URL), and `description` (a short label).

Usage

```
ecoreg_list
```

Format

A list with each element containing:

filename The name to save the downloaded file as.

link The URL to download the file.

description A short description of the file.

Value

A named list of length 4, one element per available ecoregion dataset, each with `filename`, `link`, and `description` fields as described in `Format`.

See Also

`get_ecoreg()` to download the datasets listed here, and `read_ecoreg()` to load one directly.

Examples

```
ecoreg_list
```

`evaluate_range` *Evaluate Range Maps Against Independent Validation Data*

Description

Compare range maps produced by `get_range()` with external validation data, such as species distribution models (SDMs) or expert-derived range maps, and summarize precision, sensitivity, specificity, and TSS.

Usage

```
evaluate_range(
  root_dir = NULL,
  valData_dir = NULL,
  ecoRM_dir = NULL,
  valData_type = NULL,
  verbose = TRUE,
  print_map = TRUE,
  mask = NULL,
  res_fact = NULL
)
```

Arguments

`root_dir`	Character. String giving the root directory that contains both the generated range maps and the validation data.
`valData_dir`	Character. String giving the directory, relative to `root_dir`, containing the validation data.
`ecoRM_dir`	Character. String giving the directory, relative to `root_dir`, containing range maps generated with `get_range()`.
`valData_type`	Character. String indicating the expected validation-data format: `"SHP"` or `"TIFF"`.
`verbose`	Logical. Should progress information be printed while the comparison is running?
`print_map`	Logical. If `TRUE`, write a PDF overlay map for each evaluated species.

mask	Optional SpatRaster. Used as a study-area mask and common comparison domain.
res_fact	Optional integer. Aggregation factor used to coarsen the native resolution before comparison.

Details

TIFF validation files must have file names matching the species names of the range maps. Shapefile-based validation data must include a column named sci_name with matching species names.

The function can optionally mask the focal study region and aggregate maps to coarser resolutions before calculating evaluation metrics, which is useful when comparing products with different native resolutions.

Value

A list with two elements: df_eval, a data frame containing per-species evaluation statistics, and overlay_list, a list of raster overlays used for plotting and inspection.

References

Pinkert, S., Sica, Y. V., Winner, K., & Jetz, W. (2023). The potential of ecoregional range maps for boosting taxonomic coverage in ecology and conservation. Ecography, 12, e06794. doi:10.1111/ecog.06794

See Also

get_range() to generate the range maps being evaluated, and cv_range() for cross-validation against the original occurrence data instead of external validation data.

Examples

```
## EcoRM evaluation at different resolutions (<10sec runtime)
root_dir  <- list.files(
    system.file(package = "gbif.range"),
    pattern = "extdata",
    full.names = TRUE
)

res5km <- evaluate_range(
    root_dir = root_dir,
    valData_dir = "SDM",
    ecoRM_dir = "EcoRM",
    verbose = TRUE,
    print_map = FALSE,
    valData_type = "TIFF",
    mask = NULL,
    res_fact = NULL
)

res10km <- evaluate_range(
    root_dir = root_dir,
```

```
    valData_dir = "SDM",
    ecoRM_dir = "EcoRM",
    verbose = TRUE,
    print_map = FALSE,
    valData_type = "TIFF",
    mask = NULL,
    res_fact = 2
)


## Extract and plot a specific overlay map
terra::plot(
    res10km$overlay_list[[1]],
    col = c("gray", "red", "blue", "purple"),
    breaks = c(-0.5, 0.5, 1.5, 2.5, 3.5),
    legend = FALSE,
    main = paste0("Species: ",
        res10km$df_eval[1, 1],"\n",
        "Precision = ",
        round(res10km$df_eval[1, "Prec_ecoRM"], digits = 2)," & ",
          "Sensitivity = ",
        round(res10km$df_eval[1, "Sen_ecoRM"], digits = 2)),
    las = 1
)

graphics::legend(
    "bottomright",
    legend = c(
      "Abs in both (TA)",
      "Pres in ecoRM only (FP)",
      "Pres in valRM only (FA)",
      "Pres in both (TP)"
    ),
    fill = c("gray", "red", "blue", "purple"),
    bg = NA,
    box.col = NA,
    inset = c(0,0.2)
)

## Compare sensitivity and precision outputs ##
  # Calculate overall performance (e.g. mean sensitivity & precision)
res5km$df_eval$Mean_SenPrec <-
    (res5km$df_eval$Sen_ecoRM + res5km$df_eval$Prec_ecoRM) / 2
res10km$df_eval$Mean_SenPrec <-
    (res10km$df_eval$Sen_ecoRM + res10km$df_eval$Prec_ecoRM) / 2

  # Combine the data frames and add a Resolution column
combined_df <- rbind(
    cbind(res5km$df_eval, Resolution = "5km"),
    cbind(res10km$df_eval, Resolution = "10km")
)

  # Convert to long format
```

```
variables <- c("Sen_ecoRM", "Prec_ecoRM", "Mean_SenPrec")
long_df <- data.frame(
    Variable = rep(variables, each = nrow(combined_df)),
    Value = unlist(combined_df[variables]),
    Resolution = rep(combined_df$Resolution, times = length(variables))
)

  # Plot boxplots using base R
graphics::boxplot(
    Value ~ Variable + Resolution,
    data = long_df,
    col = c("#FFC300", "#619CFF"),
    names = rep(variables, 2),
    xlab = "Variable",
    ylab = "Value",
    las = 1,
    main = "Boxplot of Sen_ecoRM and Prec_ecoRM"
)

  # Adding legend for colors
graphics::legend(
    "bottomright",
    legend = c("5km", "10km"),
    fill = c("#FFC300", "#619CFF"),
    title = "Resolution",
    bty = "n"
)
```

`fig_label` *Draw a Figure Label at a Standard Position*

Description

Draw a Figure Label at a Standard Position

Usage

```
fig_label(
  text,
  region = "figure",
  pos = "topleft",
  cex = NULL,
  margin = 0.02,
  ...
)
```

Arguments

`text`	Label text.

region	Character string specifying whether the label should be placed relative to the "figure", "plot", or "device".
pos	Character string giving the label position.
cex	Optional text expansion factor.
margin	Numeric margin used to offset the label from the selected boundary.
...	Additional arguments passed to graphics::text().

Value

Invisibly, a numeric vector c(x, y) giving the x and y boundary coordinates of the selected region used to position the label. Called primarily for its side effect of drawing the label on the current plot device.

---

getGBIF *Constructor for* getGBIF *Objects*

Description

Wrap a data frame in the getGBIF class used by downstream functions in the package.

Usage

```
getGBIF(df)
```

Arguments

df	A data frame containing GBIF occurrence records.

Value

An object of class c("getGBIF", "data.frame").

---

getRange-class *Reference Class for* get_range() *Results*

Description

Stores the original arguments used to build a range map and the resulting spatial output.

Value

A generator object of reference class "getRange", used to instantiate objects with two fields: init.args (the original arguments used to build the range map) and rangeOutput (the resulting spatial output).

get_doi *Create a DOI for GBIF-derived datasets*

Description

A lightweight wrapper around `rgbif::derived_dataset()` that combines one or more `get_gbif()` outputs into a citable derived dataset.

Usage

```
get_doi(
  gbifs = NULL,
  title = NULL,
  description = NULL,
  source_url = "https://example.com/",
  user = "",
  pwd = "",
  ...
)
```

Arguments

`gbifs`	`getGBIF` object or a list of `getGBIF` objects.
`title`	Character. String giving the title of the derived dataset.
`description`	Character. String describing the dataset.
`source_url`	Character. String pointing to the location where the derived dataset or workflow is documented.
`user`	Character. String with the GBIF username used to submit the request.
`pwd`	Character. String with the GBIF password used to submit the request.
`...`	Additional arguments passed to `rgbif::derived_dataset()`.

Details

See `rgbif::derived_dataset()` for GBIF requirements and the structure of the returned metadata.

Value

The object returned by `rgbif::derived_dataset()`, including the DOI request metadata generated by GBIF.

References

Chamberlain, S., Oldoni, D., & Waller, J. (2022). rgbif: interface to the global biodiversity information facility API. doi:10.5281/zenodo.6023735

See Also

get_gbif() to produce the getGBIF objects used as input here; the **rgbif** package for more general workflows to create GBIF derived datasets and DOIs.

Examples

```
## Not run:  # unknown credentials
# Download worldwide observations for two species
obs_ps <- get_gbif("Phocoena sinus")
obs_am <- get_gbif("Ailuropoda melanoleuca")

# Retrieve a DOI for one get_gbif() output
get_doi(
       gbifs = obs_ps,
       title = "GBIF_test1",
       description = "A small example 1",
       source_url = "https://example.com/",
       user = "",
       pwd = ""
) # Use your own GBIF credentials here

# Retrieve a DOI for several get_gbif() outputs
get_doi(
       gbifs = list(obs_ps, obs_am),
       title = "GBIF_test2",
       description = "A small example 2",
       source_url = "https://example.com/",
       user = "",
       pwd = ""
) # Use your own GBIF credentials here


## End(Not run)
```

---

get_ecoreg *Download Ecoregion Layers*

Description

Download one or more ecoregion datasets listed in ecoreg_list() and unpack them into a local directory.

Usage

```
get_ecoreg(ecoreg_name = "all", save_dir = NULL)
```

Arguments

ecoreg_name	Character. Use "all" to download every dataset, or supply a single file name listed in ecoreg_list.
save_dir	Character. Directory where the downloaded zip files and extracted shapefiles should be stored. Defaults to tempdir() unless the gbif.range.save_dir option or GBIF_RANGE_SAVE_DIR environment variable is set, in which case that location is used instead.

Value

NULL. Files are downloaded and unpacked for side effects.

See Also

ecoreg_list for the list of available datasets, and read_ecoreg() to download and load a layer in one step.

Examples

```
# Download one of the ecoregion datasets listed in ecoreg_list
get_ecoreg(ecoreg_name = "eco_marine", save_dir = tempdir())
```

---

get_gbif *Download and Filter GBIF Occurrences for Spatial Analyses*

Description

Downloads occurrence records from GBIF for a focal taxon, resolving the input name against the GBIF backbone taxonomy and querying by accepted taxon key — thereby capturing records linked to synonyms and infra-specific taxa (subspecies, varieties) in the same way as the GBIF website. Large requests are automatically split into spatial tiles, and a series of post-download filters can be applied to clean records for spatial analyses and range mapping.

Usage

```
get_gbif(
  sp_name = NULL,
  search = TRUE,
  rank = NULL,
  phylum = NULL,
  class = NULL,
  order = NULL,
  family = NULL,
  conf_match = 80,
  geo = NULL,
  has_xy = TRUE,
```

```
  spatial_issue = FALSE,
  grain = 100,
  duplicates = FALSE,
  absences = FALSE,
 basis = c("OBSERVATION", "HUMAN_OBSERVATION", "MACHINE_OBSERVATION", "OCCURRENCE",
    "MATERIAL_CITATION", "MATERIAL_SAMPLE", "LITERATURE"),
 establishment = c("native", "casual", "released", "reproducing", "established",
    "colonising", "invasive", "widespreadInvasive"),
  add_infos = NULL,
  time_period = c(1000, 3000),
  identic_xy = FALSE,
  wConverted_xy = TRUE,
  centroids = FALSE,
  ntries = 10,
  error_skip = TRUE,
  occ_samp = 10000,
  should_use_occ_download = FALSE,
  occ_download_user = NULL,
  occ_download_pwd = NULL,
  occ_download_email = NULL,
  verbose = TRUE,
  ...
)
```

Arguments

sp_name	Character. String with the species name. Scientific names at genus-species level are expected; fuzzy matching is available when search = FALSE.
search	Logical. If TRUE (default), use a strict GBIF backbone search and keep only species-, subspecies-, or variety-level matches. If FALSE, use a more permissive search and optionally rely on rank, phylum, class, order, and family to resolve ambiguous matches.
rank	Character. String giving the preferred rank to keep: "SPECIES", "SUBSPECIES", or "VARIETY". When NULL, rank priority is inferred from sp_name.
phylum	Optional character. Phylum used to disambiguate alternative GBIF matches. Particularly useful for hemihomonyms.
class	Optional character. Class used to disambiguate alternative GBIF matches.
order	Optional character. Order used to disambiguate alternative GBIF matches.
family	Optional character. Family used to disambiguate alternative GBIF matches.
conf_match	Numeric. Confidence threshold between 0 and 100 for the GBIF backbone match. Default is 80.
geo	Spatial object. Used to restrict the query extent. Accepted classes are Extent, SpatExtent, SpatialPolygon, SpatialPolygonDataFrame, SpatVector, and sf. The default NULL queries the whole globe.
has_xy	Logical. If TRUE (default), keep only records with coordinates. If FALSE, keep only records without coordinates. If NULL, keep all records.

spatial_issue Logical. If FALSE (default), keep only records without geospatial issues. If TRUE, keep only records with geospatial issues. If NULL, keep all records.

grain Numeric. Study grain in kilometers. Default is 100. The value is used to filter records by coordinateUncertaintyInMeters and by the number of reported coordinate decimals.

duplicates Logical. Should duplicate records be kept? Default is FALSE.

absences Logical. Should absence records be kept? Default is FALSE.

basis Character. Vector of accepted basisOfRecord values. Default excludes specimen-based records (e.g. herbarium sheets, museum specimens) and records of unknown basis. See **Details** for the full list of accepted values, and a note on why "MATERIAL_SAMPLE" can be misleading.

establishment Character. Vector of accepted degreeOfEstablishment values. Default keeps native and broadly established records; records lacking this field are always kept regardless of the selected values. See **Details** for the full list of accepted values.

add_infos Optional character. Additional GBIF occurrence fields to append to the default output columns. See **Details** for the default columns and where to find all available GBIF field names.

time_period Numeric. Year range to retain. Default is c(1000, 3000). Records with missing years are kept.

identic_xy Logical. Should records with identical longitude-latitude pairs be kept? Default is FALSE.

wConverted_xy Logical. Should records that appear to be incorrectly converted from degree-minute notation be kept? Default is TRUE. If FALSE, an approximate version of CoordinateCleaner::cd_ddmm() is applied.

centroids Logical. Should records located on raster centroids be kept? Default is FALSE. If FALSE, CoordinateCleaner::cd_round() is used.

ntries Numeric. Number of download attempts before giving up after GBIF or rgbif errors. Default is 10.

error_skip Logical. If TRUE, return an empty result when all download attempts fail.

occ_samp Numeric. Maximum number of GBIF occurrences sampled per geographic tile. Default is 10000.

should_use_occ_download
Logical. If TRUE, use rgbif::occ_download() instead of rgbif::occ_search(). This requires GBIF credentials. Default is FALSE.

occ_download_user
Character. GBIF username. Required if should_use_occ_download = TRUE.

occ_download_pwd
Character. GBIF password. Required if should_use_occ_download = TRUE.

occ_download_email
Character. GBIF email address. Required if should_use_occ_download = TRUE.

verbose Logical. Should progress messages be printed? Default is TRUE.

... Additional arguments passed to CoordinateCleaner::cd_round().

Details

The function follows the same taxonomic matching logic used by the GBIF website. Internally, the input name is first resolved against the GBIF backbone taxonomy. If the matched name is a synonym, the download uses the corresponding accepted GBIF taxon key rather than the synonym key itself.

Querying by accepted taxon key means that occurrence records linked to infra-specific taxa (subspecies, varieties) under that accepted name are also retrieved, mirroring GBIF website behaviour. The `acceptedTaxonKey` and `scientificName` columns in the output therefore reflect the original GBIF record-level assignment and may refer to a subspecies or variety rather than the species-level input name. Use `get_status(children = TRUE)` beforehand to inspect which infra-specific keys fall under the queried taxon concept.

The output preserves the original GBIF taxonomic fields for each record rather than rewriting them to a single accepted name. In particular, `scientificName` stores the record-level GBIF name, while `acceptedScientificName`, `acceptedTaxonKey`, and `taxonomicStatus` reflect the harmonized taxon concept used for the query.

basis: available values (old and new GBIF vocabulary) are `"OBSERVATION"`, `"HUMAN_OBSERVATION"`, `"MACHINE_OBSERVATION"`, `"MATERIAL_CITATION"`, `"MATERIAL_SAMPLE"`, `"PRESERVED_SPECIMEN"`, `"FOSSIL_SPECIMEN"`, `"LIVING_SPECIMEN"`, `"LITERATURE"`, `"UNKNOWN"`, and `"OCCURRENCE"`. The default setting removes specimen-based records and records of unknown basis. See (here) for a description of each category.

Records linked to non-taxonomic backbone entries (e.g., BOLD barcode sequences) are typically registered under `basisOfRecord = "MATERIAL_SAMPLE"`. These sequence-based, bulk, or environmental DNA (eDNA) records may reflect the physical collection site rather than the precise microhabitat. However, this spatial imprecision is not critical as the package is aimed to operate at macro-ecological scales where local site-level offsets do not alter the broader range estimations.

establishment: available `degreeOfEstablishment` values are `"native"`, `"casual"`, `"released"`, `"reproducing"`, `"established"`, `"colonising"`, `"invasive"`, `"widespreadInvasive"`, `"managed"`, `"captive"`, `"cultivated"`, `"unestablished"`, and `"failing"`. The default keeps native and broadly established records; records with no `degreeOfEstablishment` information are always kept regardless of the selected values. See (here) for a description of each category.

add_infos: the default output always includes `"taxonKey"`, `"scientificName"`, `"acceptedTaxonKey"`, `"acceptedScientificName"`, `"individualCount"`, `"decimalLatitude"`, `"decimalLongitude"`, `"basisOfRecord"`, `"coordinateUncertaintyInMeters"`, `"countryCode"`, `"country"`, `"year"`, `"datasetKey"`, `"institutionCode"`, `"publishingOrgKey"`, `"taxonomicStatus"`, `"taxonRank"`, and `"degreeOfEstablishment"`. Any additional field name accepted by the GBIF occurrence API can be requested via `add_infos`; see the full list (here).

If the requested extent contains many records, the query is fragmented into spatial tiles so that individual API calls remain manageable. Tiles are refined until each contains fewer than 10,000 records, which is faster and more robust than relying on a single large request.

Post-download filtering can remove records outside the study extent, duplicated coordinates, absences, unwanted bases of record, records with low spatial precision, suspicious coordinate conversions, and raster-centroid records. The `grain` argument controls both coordinate-uncertainty filtering and the minimum number of coordinate decimals that must be reported.

Decimal filtering follows these thresholds:

- if 110 km > `grain` $\geq$ 11 km, coordinates with at least 1 decimal are kept
- if 11 km > `grain` $\geq$ 1100 m, coordinates with at least 2 decimals are kept
- if 1100 m > `grain` $\geq$ 110 m, coordinates with at least 3 decimals are kept
- if 110 m > `grain` $\geq$ 11 m, coordinates with at least 4 decimals are kept
- if 11 m > `grain` $\geq$ 1.1 m, coordinates with at least 5 decimals are kept

Value

A `getGBIF` object (a data.frame subclass) containing the requested occurrence data. The object may also carry the attributes `filter_log`, which records how many rows were removed at each filtering step, and `no_xy`, which stores records without coordinates when they are requested.

Taxonomic harmonization is reflected in the output columns rather than by replacing all names with a single accepted label. In particular, `scientificName` stores the record-level GBIF name, whereas `acceptedScientificName`, `acceptedTaxonKey`, and `taxonomicStatus` expose the accepted GBIF interpretation used by the query.

References

Chauvier, Y., Thuiller, W., Brun, P., Lavergne, S., Descombes, P., Karger, D. N., ... & Zimmermann, N. E. (2021). Influence of climate, soil, and land cover on plant species distribution in the European Alps. Ecological Monographs, 91(2), e01433. doi:10.1002/ecm.1433

Chamberlain, S., Oldoni, D., & Waller, J. (2022). rgbif: interface to the global biodiversity information facility API. doi:10.5281/zenodo.6023735

Zizka, A., Silvestro, D., Andermann, T., Azevedo, J., Duarte Ritter, C., Edler, D., ... & Antonelli, A. (2019). CoordinateCleaner: Standardized cleaning of occurrence records from biological collection databases. Methods in Ecology and Evolution, 10(5), 744-751. doi:10.1111/2041210X.13152

Hijmans, R. J. (2022). terra: Spatial Data Analysis. R package version 1.6-7. https://CRAN.R-project.org/package=terra

See Also

`get_status`() to inspect the accepted name and synonym mapping returned by GBIF; the **rgbif** package for more general GBIF retrieval workflows; and the **CoordinateCleaner** package for more extensive occurrence cleaning.

Examples

```
# Download all worldwide observations of the great panda, with:
# - 100km grain
# - after 1990
# - keeping duplicates
# - adding the name of the person who collected the panda records
obs_am <- get_gbif(
```

```
       sp_name = "Ailuropoda melanoleuca",
       grain = 100,
       duplicates = TRUE,
       time_period = c(1990,3000),
       add_infos = c("recordedBy","issue")
)

# Guard against an unavailable remote source
if (gbif_have(obs_am)) {

# Extract borders
countries <- terra::vect(
  system.file("extdata", "world_countries.shp", package = "gbif.range")
)

# Plot
terra::plot(countries, col = "#bcbddc")
graphics::points(
       obs_am[,c("decimalLongitude","decimalLatitude")],
       pch = 20,
       col = "#238b4550",
       cex = 4
)

}
```

`get_gbif_count` *Count GBIF Occurrences Before Downloading Data*

Description

Query GBIF and return the number of available records for a taxon using the same name-matching logic as `get_gbif()`, but without downloading the occurrence table.

Usage

```
get_gbif_count(
  sp_name = NULL,
  search = TRUE,
  rank = NULL,
  phylum = NULL,
  class = NULL,
  order = NULL,
  family = NULL,
  conf_match = 80,
  geo = NULL,
  has_xy = TRUE,
  spatial_issue = FALSE,
```

```
  verbose = TRUE
)
```

Arguments

sp_name	Character. String with the species name. Scientific names at genus-species level are expected; fuzzy matching is available when search = FALSE.
search	Logical. If TRUE (default), use a strict GBIF backbone search and keep only species-, subspecies-, or variety-level matches. If FALSE, use a more permissive search and optionally rely on rank, phylum, class, order, and family to resolve ambiguous matches.
rank	Character. String giving the preferred rank to keep: "SPECIES", "SUBSPECIES", or "VARIETY". When NULL, rank priority is inferred from sp_name.
phylum	Optional character. Phylum used to disambiguate alternative GBIF matches. Particularly useful for hemihomonyms.
class	Optional character. Class used to disambiguate alternative GBIF matches.
order	Optional character. Order used to disambiguate alternative GBIF matches.
family	Optional character. Family used to disambiguate alternative GBIF matches.
conf_match	Numeric. Confidence threshold between 0 and 100 for the GBIF backbone match. Default is 80.
geo	Spatial object. Used to restrict the query extent. Accepted classes are Extent, SpatExtent, SpatialPolygon, SpatialPolygonDataFrame, SpatVector, and sf. The default NULL queries the whole globe.
has_xy	Logical. If TRUE (default), count only records with coordinates. If FALSE, count only records without coordinates. If NULL, count all records.
spatial_issue	Logical. If FALSE (default), count only records without geospatial issues. If TRUE, count only records with geospatial issues. If NULL, count all records.
verbose	Logical. Should the formatted summary be printed to the console? Default is TRUE.

Details

The function mirrors the taxonomic matching strategy used by get_gbif(), then reports both the total number of GBIF records and the number retained after applying the chosen filters.

Value

A numeric vector of length two giving the total number of GBIF records and the number retained by the requested filters. If verbose = TRUE, a formatted summary is also printed to the console.

References

Chamberlain, S., Oldoni, D., & Waller, J. (2022). rgbif: interface to the global biodiversity information facility API. doi:10.5281/zenodo.6023735

See Also

get_gbif() to download the occurrence records counted by this function.

Examples

```
# Get number of observations with default filters
get_gbif_count(
sp_name = "Ailuropoda melanoleuca",
has_xy = TRUE,
spatial_issue = FALSE,
geo = NULL
)

# Get the total number of observations
get_gbif_count(
sp_name = "Ailuropoda melanoleuca",
has_xy = NULL,
spatial_issue = NULL,
geo = NULL
)

# Example of setting global 'geo' (all records are still kept)
get_gbif_count(
sp_name = "Ailuropoda melanoleuca",
has_xy = NULL,
spatial_issue = NULL,
geo = terra::ext()
)

# Example of fuzzy matching when search is set to FALSE
get_gbif_count(
sp_name = "Ailuropoda melanolca",
search = FALSE
)

# Example on the European Alps (has_xy = TRUE by default)
shp.lonlat <- terra::vect(
    paste0(
        system.file(package = "gbif.range"),
        "/extdata/shp_lonlat.shp"
    )
)
get_gbif_count(
sp_name = "Arctostaphylos alpinus",
has_xy = TRUE,
spatial_issue = FALSE,
geo = shp.lonlat
)
```

get_range *Create Species Range Maps from Occurrences and Ecoregions*

Description

Estimate ecologically informed species ranges from occurrence data and an ecoregion layer. The workflow combines outlier filtering, clustering, convex hull construction, and intersection with occupied ecoregions.

Usage

```
get_range(
  occ_coord = NULL,
  ecoreg = NULL,
  ecoreg_name = NULL,
  degrees_outlier = 5,
  clust_pts_outlier = 4,
  buff_width_point = 4,
  buff_incrmt_pts_line = 0.5,
  buff_width_polygon = 4,
  dir_temp = tempdir(),
  format = c("SpatVector", "sf", "SpatRaster"),
  res = 0.1,
  verbose = TRUE
)
```

Arguments

occ_coord	getGBIF object returned by get_gbif() or a data.frame containing the columns decimalLongitude and decimalLatitude.
ecoreg	Spatial ecoregion layer in WGS84. Accepted classes are SpatialPolygonsDataFrame, SpatVector, and sf. This can be a downloaded layer from read_ecoreg() or a custom layer created by make_ecoreg().
ecoreg_name	Character. String naming the field in ecoreg that defines ecoregion categories. For eco_terra, for example, "ECO_NAME" is the most detailed level. When ecoreg comes from make_ecoreg(), "EcoRegion" is used automatically.
degrees_outlier	Numeric. Distance threshold in degrees. Points whose clust_pts_outlier-th nearest neighbour lies beyond this distance are treated as outliers. Default is 5.
clust_pts_outlier	Numeric. k-nearest-neighbour order used for outlier detection. Default is 4.
buff_width_point	Numeric. Buffer width in degrees for isolated points.
buff_incrmt_pts_line	Numeric. Increment in buffer width for linear clusters.

`buff_width_polygon`
: Numeric. Buffer width in degrees applied to convex hull polygons.

`dir_temp`
: Character. String giving the parent directory used for temporary convex-hull files. A uniquely named sub-directory is created inside it and removed when the function exits; `dir_temp` itself is never deleted. Defaults to `tempdir()`.

`format`
: Character. Output format for the range map. One of `"SpatVector"` (default), `"sf"`, or `"SpatRaster"`. `"SpatRaster"` rasterizes the range at the resolution set by `res`.

`res`
: Numeric. Output resolution in degrees when `format = "SpatRaster"`. Default is `0.1` (about 11.1 km at the equator). The achievable resolution is constrained by the spatial precision of `ecoreg`.

`verbose`
: Logical. Should progress messages be printed?

Details

The function follows four main steps.

First, occurrence points are filtered for spatial outliers using k-nearest-neighbour distances and are assigned to ecoregions.

Second, points within occupied ecoregions are grouped into clusters using a combination of Gaussian mixture modeling and k-means clustering.

Third, each cluster is converted into a polygon. Single points receive circular buffers, collinear clusters receive line-based buffers, and other clusters receive buffered convex hulls through `conv_function()`.

Fourth, cluster polygons are intersected with their parent ecoregions and merged into a final range layer.

Download-ready ecoregion datasets include `eco_terra` for terrestrial species, `eco_fresh` for freshwater species, and `eco_marine` or `eco_hd_marine` for marine species.

Value

An object of class `getRange` with two fields: `init.args`, containing the arguments and data used to build the map, and `rangeOutput`, containing the resulting `SpatVector`, `sf`, or `SpatRaster` object.

For `format = "SpatVector"` and `format = "sf"`, the range is returned as **one feature per occupied ecoregion** rather than a single merged polygon. Features may be multipart where a species occupies several disjoint patches of the same ecoregion; `terra::disagg()` separates them. Because ecoregions do not overlap, the features do not overlap either, so their areas are additive and `sum(terra::expanse(x))` gives the total range size. Each feature carries the `ecoreg_name` field it was clipped to plus a `species` column. Use `merge_range()` to dissolve the features into one polygon, or `format = "SpatRaster"` for a gridded range.

References

Hagen, O., Vaterlaus, L., Albouy, C., Brown, A., Leugger, F., Onstein, R. E., Novaes de Santana, C., Scotese, C. R., & Pellissier, L. (2019). Mountain building, climate cooling and the richness of cold-adapted plants in the Northern Hemisphere. Journal of Biogeography. doi:10.1111/jbi.13653

Lyu, L., Leugger, F., Hagen, O., Fopp, F., Boschman, L. M., Strijk, J. S., ... & Pellissier, L. (2022). An integrated high resolution mapping shows congruent biodiversity patterns of Fagales and Pinales. New Phytologist, 235(2), 759-772. doi:10.1111/nph.18158

Olson, D. M., Dinerstein, E., Wikramanayake, E. D., Burgess, N. D., Powell, G. V. N., Underwood, E. C., D'Amico, J. A., Itoua, I., Strand, H. E., Morrison, J. C., Loucks, C. J., Allnutt, T. F., Ricketts, T. H., Kura, Y., Lamoreux, J. F., Wettengel, W. W., Hedao, P., Kassem, K. R. (2001). Terrestrial ecoregions of the world: a new map of life on Earth. BioScience, 51(11), 933-938. doi:10.1641/00063568(2001)051[0933:TEOTWA]2.0.CO;2

The Nature Conservancy (2009). Global Ecoregions, Major Habitat Types, Biogeographical Realms and The Nature Conservancy Terrestrial Assessment Units. GIS layers developed by The Nature Conservancy with multiple partners, combined from Olson et al. (2001), Bailey 1995 and Wiken 1986. Cambridge (UK): The Nature Conservancy.

Spalding, M. D., Fox, H. E., Allen, G. R., Davidson, N., Ferdaña, Z. A., Finlayson, M., Halpern, B. S., Jorge, M. A., Lombana, A., Lourie, S. A., Martin, K. D., McManus, E., Molnar, J., Recchia, C. A., Robertson, J. (2007). Marine Ecoregions of the World: A Bioregionalization of Coastal and Shelf Areas. BioScience, 57(7), 573-583. doi:10.1641/B570707

Spalding, M. D., Agostini, V. N., Rice, J., & Grant, S. M. (2012). Pelagic provinces of the world: a biogeographic classification of the world's surface pelagic waters. Ocean & Coastal Management, 60, 19-30. doi:10.1016/j.ocecoaman.2011.12.016

The Nature Conservancy (2012). Marine Ecoregions and Pelagic Provinces of the World. GIS layers developed by The Nature Conservancy with multiple partners, combined from Spalding et al. (2007) and Spalding et al. (2012). Cambridge (UK): The Nature Conservancy.

Abell, R., Thieme, M. L., Revenga, C., Bryer, M., Kottelat, M., Bogutskaya, N., Coad, B., Mandrak, N., Contreras Balderas, S., Bussing, W., Stiassny, M. L. J., Skelton, P., Allen, G. R., Unmack, P., Naseka, A., Ng, R., Sindorf, N., Robertson, J., Armijo, E., Higgins, J. V., Heibel, T. J., Wikramanayake, E., Olson, D., Lopez, H. L., Reis, R. E., Lundberg, J. G., Sabaj Perez, M. H., Petry, P. (2008). Freshwater Ecoregions of the World: A New Map of Biogeographic Units for Freshwater Biodiversity Conservation. BioScience, 58(5), 403-414. doi:10.1641/B580507

Hijmans, R. J. (2022). terra: Spatial Data Analysis. R package version 1.6-7. https://CRAN.R-project.org/package=terra

See Also

read_ecoreg() and make_ecoreg() to prepare the ecoregion layer used here; cv_range() and evaluate_range() to evaluate the resulting range map; merge_range() to dissolve the returned polygons into a single range polygon.

Examples

```
# Load available ecoregions
eco_terra <- read_ecoreg(
    ecoreg_name = "eco_terra",
    save_dir = tempdir()
)

# First download the whole available observations of the panda from GBIF
obs_am <- get_gbif(sp_name = "Ailuropoda melanoleuca")

# Guard against an unavailable remote source
if (gbif_have(eco_terra, obs_am)) {
```

```
# Build a range map from occurrence points
range_panda <- get_range(
    occ_coord = obs_am,
    ecoreg = eco_terra,
    ecoreg_name = "ECO_NAME",
    format = "SpatRaster"
)

# Plot
    # Plot political world boundaries
countries <- terra::vect(
  system.file("extdata", "world_countries.shp", package = "gbif.range")
)
terra::plot(
    terra::crop(countries, terra::ext(73.0, 135.0, 18.0, 54.0)),
    col = "#bcbddc"
)

    # Plot range
terra::plot(
    range_panda$rangeOutput,
    axes = FALSE,
    box = FALSE,
    legend = FALSE,
    col = "chartreuse4",
    main = paste("Range:", obs_am$scientificName[1]),
    add = TRUE
)

    # Plot the occurrence points
graphics::points(
    obs_am[, c("decimalLongitude","decimalLatitude")],
    pch = 20,
    col = "#99340470",
    cex = 1.5
)

}
```

`get_status` *Retrieve Taxonomy and IUCN Status from GBIF*

Description

Query the GBIF backbone taxonomy for a species name, report the matched accepted name and synonyms, and return the associated IUCN status when available.

Usage

```
get_status(
  sp_name = NULL,
  search = TRUE,
  rank = NULL,
  phylum = NULL,
  class = NULL,
  order = NULL,
  family = NULL,
  conf_match = 80,
  level = c("accepted", "children", "all"),
  verbose = TRUE
)
```

Arguments

sp_name	Character. String with the species name. Scientific names at genus-species level are expected; fuzzy matching is available when search = FALSE.
search	Logical. If TRUE (default), use a strict GBIF backbone search and keep only species-, subspecies-, or variety-level matches. If FALSE, use a more permissive search and optionally rely on rank, phylum, class, order, and family to resolve ambiguous matches.
rank	Character. String giving the preferred rank to keep: "SPECIES", "SUBSPECIES", or "VARIETY". When NULL, rank priority is inferred from sp_name.
phylum	Optional character. Phylum used to disambiguate alternative GBIF matches. Particularly useful for hemihomonyms.
class	Optional character. Class used to disambiguate alternative GBIF matches.
order	Optional character. Order used to disambiguate alternative GBIF matches.
family	Optional character. Family used to disambiguate alternative GBIF matches.
conf_match	Numeric. Confidence threshold between 0 and 100 for the GBIF backbone match. Default is 80.
level	Character. Controls how much of the GBIF taxon concept is returned. "accepted" (default) returns the accepted name and its synonyms. "children" additionally includes infra-specific taxa (subspecies, varieties) as CHILDREN rows, mirroring the scope of get_gbif(). "all" further adds alternative name string representations (orthographic variants, etc.) as RELATED rows, which are not used by get_gbif() and are intended for taxonomic inspection only.
verbose	Logical. Should status messages be printed to the console when no match is found? Default is TRUE.

Details

Use get_status() before get_gbif() when you want to inspect how GBIF resolves an input name, which name is treated as the accepted backbone taxon, and which synonyms are included in the taxon concept used for occurrence retrieval.

When `level = "accepted"`, the returned rows correspond to the accepted name and synonyms linked to the accepted GBIF taxon key. When `level = "children"`, infra-specific taxa that `get_gbif()` also retrieves are added, giving a complete picture of the occurrence download scope. When `level = "all"`, alternative name representations are further included for taxonomic review.

IUCN Red List status is retrieved from GBIF for the accepted taxon key only. GBIF does not currently store subspecies-level IUCN assessments, so the `IUCN_status` column is uniform across all rows. Note that some subspecies have independent IUCN assessments (e.g. *Panthera tigris sumatrae* is Critically Endangered) that are not reflected here; consult the IUCN Red List directly for infra-specific status.

Value

A data frame with the columns `canonicalName`, `rank`, `gbif_key`, `scientificName`, `gbif_status`, `Genus`, `Family`, `Order`, `Class`, `Phylum`, `IUCN_status`, and `sp_nameMatch`. The row with `gbif_status = "ACCEPTED"` identifies the accepted GBIF taxon concept used by `get_gbif()`, while `sp_nameMatch` marks the row that most closely matches the submitted input name. Additional rows carry `gbif_status = "CHILDREN"` (infra-specific taxa, when `level = "children"` or `level = "all"`) or `gbif_status = "RELATED"` (alternative name representations, when `level = "all"`).

References

Chamberlain, S., Oldoni, D., & Waller, J. (2022). rgbif: interface to the global biodiversity information facility API. doi:10.5281/zenodo.6023735

See Also

`get_gbif()` to download occurrences for the accepted taxon concept returned here; the **rgbif** package for more general approaches to querying the GBIF backbone taxonomy.

Examples

```
# --- 1. Default: accepted name + synonyms only ---
tax <- get_status("Ailuropoda melanoleuca")
tax

# --- 2. Include infra-specific taxa (subspecies, varieties) ---
# These are the same keys retrieved by get_gbif()
tax_ch <- get_status("Ailuropoda melanoleuca", level = "children")
tax_ch

# --- 3. Include alternative name string representations (inspection only) ---
# RELATED rows are not used by get_gbif()
tax_rel <- get_status("Ailuropoda melanoleuca", level = "all")
tax_rel

# --- 4. Fuzzy matching for uncertain names ---
# sp_nameMatch = "VARIANT" for close but non-identical fuzzy matches
get_status("Ailuropoda melanoleuca", search = FALSE)

# --- 5. Cross-check get_status() keys against get_gbif() output ---
occ <- get_gbif("Ailuropoda melanoleuca", has_xy = TRUE, verbose = FALSE)
```

```
# Guard against an unavailable remote source
if (gbif_have(tax_ch, occ)) {

valid_keys    <- tax_ch$gbif_key[tax_ch$gbif_status %in% c("ACCEPTED", "CHILDREN")]
returned_keys <- unique(occ$acceptedTaxonKey)

# Keys in get_status() - backbone taxa (ACCEPTED + CHILDREN)
valid_keys

# Keys returned by get_gbif()
returned_keys

# Are all occurrence keys known backbone keys?
# FALSE indicates non-backbone entries (e.g. BOLD sequences) are present
all(returned_keys %in% valid_keys)

# Inspect non-backbone records - typically MATERIAL_SAMPLE, excluded by default
extra_keys <- returned_keys[!returned_keys %in% valid_keys]
extra_keys
occ[occ$acceptedTaxonKey %in% extra_keys,
    c("acceptedTaxonKey", "scientificName", "basisOfRecord")]

# Note: not all valid_keys need to appear in returned_keys - some subspecies
# may have no records in GBIF at all (e.g. extinct taxa) or may have been
# removed by get_gbif() filtering options (e.g. has_xy, basis, grain)

# --- 6. Input name flagged correctly regardless of how GBIF resolves it ---
# sp_nameMatch = "INPUT" marks the row closest to the submitted name
tax_ch[tax_ch$sp_nameMatch == "INPUT", ]

}
```

`make_blocks` *Split Data into Approximately Balanced Folds*

Description

Create a fold-assignment vector for random or spatially structured cross-validation.

Usage

```
make_blocks(
  nfolds = 5,
  df = data.frame(),
  nblocks = nfolds * 2,
  npoints = NA,
  pres = numeric()
)
```

Arguments

nfolds	Numeric. Number of folds to create.
df	Optional data.frame. Columns define the structure used to build folds. When omitted, random folds are created from npoints.
nblocks	Numeric. Number of initial clusters used to build the folds. Must be at least nfolds.
npoints	Optional numeric. Number of observations to split when df is not supplied.
pres	Optional binary vector. Used to restrict CLARA clustering to a subset of rows and assign the remainder by nearest neighbours.

Details

If df has one column, folds are based on quantile bins. If df has two or more columns, folds are based on CLARA clustering. Remaining clusters are assigned to folds with an optimization step that tries to balance fold sizes as evenly as possible.

Value

An integer vector of length nrow(df) or npoints, giving the fold assignment for each observation.

References

Brun, P., Thuiller, W., Chauvier, Y., Pellissier, L., Wüest, R. O., Wang, Z., & Zimmermann, N. E. (2020). Model complexity affects species distribution projections under climate change. Journal of Biogeography, 47(1), 130-142. doi:10.1111/jbi.13700

See Also

cv_range() which uses this function internally to create cross-validation folds.

Examples

```
# Downloading worldwide all the observations of the great panda
obs_am <- get_gbif(sp_name = "Ailuropoda melanoleuca")

# Guard against an unavailable remote source
if (gbif_have(obs_am)) {

# Create a vector of folds (n = 5) spatially blocked (n = 10)
block_am <- make_blocks(
    nfolds = 5,
    df = obs_am[, c("decimalLatitude","decimalLongitude")],
    nblocks = 5
)

# Plot one colour per fold
countries <- terra::vect(
  system.file("extdata", "world_countries.shp", package = "gbif.range")
)
```

```
countries_focus <- terra::crop(
    countries,
    terra::ext(73.0, 135.0, 18.0, 54.0)
)
terra::plot(countries_focus, col = "#bcbddc")
graphics::points(
    obs_am[, c("decimalLongitude","decimalLatitude")],
    pch = 20,
    col = block_am,
    cex = 1
)

}
```

---

make_ecoreg *Build Custom Ecoregions from Environmental Layers*

Description

Cluster multi-layer environmental data to create a custom ecoregion map that can be used directly in `get_range()`.

Usage

```
make_ecoreg(
  env = NULL,
  nclass = NULL,
  path = "",
  name = "",
  format = c("SpatVector", "sf", "SpatRaster"),
  verbose = TRUE,
  ...
)
```

Arguments

`env`	Raster stack. Can be any abiotic or biotic factors thought of defining ecoregions boundaries. Accepted classes are `SpatRaster`, `RasterBrick`, and `RasterStack`.
`nclass`	Numeric. Number of environmental classes to create.
`path`	Optional character. Directory where the output should be written. Leave empty to return the result directly.
`name`	Character. Output file name without extension when `path` is used.
`format`	Output format. One of `"SpatVector"` (default), `"sf"`, or `"SpatRaster"`. `"SpatRaster"` returns the raw cluster raster instead of converting to polygons.
`verbose`	Logical. Should progress messages be printed? Default is `TRUE`.
`...`	Additional arguments passed to `cluster::clara()`.

Details

This function is useful when the packaged ecoregion layers are too coarse for a study area or when a custom environmental regionalization is needed. Clusters are created with the CLARA algorithm on the multivariate environmental space represented by `env`.

Value

If `path == ""`, returns the generated raster or polygon object. Otherwise, writes the output to disk as a GeoTIFF or Shapefile.

References

Chauvier, Y., Zimmermann, N. E., Poggiato, G., Bystrova, D., Brun, P., & Thuiller, W. (2021). Novel methods to correct for observer and sampling bias in presence-only species distribution models. Global Ecology and Biogeography, 30(11), 2312-2325. doi:10.1111/geb.13383

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., & Hornik, K. (2021). cluster: Cluster Analysis Basics and Extensions. R package version 2.1.2. https://CRAN.R-project.org/package=cluster/

Reynolds, A. P., Richards, G., de la Iglesia, B., & Rayward-Smith, V. J. (2006). Clustering rules: A comparison of partitioning and hierarchical clustering algorithms. Journal of Mathematical Modelling and Algorithms, 5(4), 475-504. doi:10.1007/s1085200590221

Schubert, E., & Rousseeuw, P. J. (2019). Faster k-Medoids clustering: Improving the PAM, CLARA, and CLARANS algorithms. In G. Amato, C. Gennaro, V. Oria, & M. Radovanović (Eds.), Similarity search and applications. SISAP 2019. Lecture Notes in Computer Science (Vol. 11807, pp. 171-187). Springer.

See Also

`get_range()` to build a range map using the ecoregion layer produced here.

Examples

```
# Open data
rst_path <- paste0(
system.file(package = "gbif.range"),
"/extdata/rst_enl.tif"
)
rst <- terra::rast(rst_path)
shp_path <- paste0(
system.file(package = "gbif.range"),
"/extdata/shp_lonlat.shp"
)
shp_lonlat <- terra::vect(shp_path)
rst <- terra::crop(rst, shp_lonlat)

# Apply the function by inferring 50 environmental classes
my_eco <- make_ecoreg(env = rst,
nclass = 50,
format = "sf"
```

```
)


# Downloading in the European Alps the observations of one plant species
obs_paed <- get_gbif(
  sp_name = "Paederota bonarota",
  geo = shp_lonlat,
  grain = 1
)

# Guard against an unavailable remote source
if (gbif_have(obs_paed)) {

# Create the range map based on:
# - custom ecoregion at 5 x 5 km resolution
# - smaller buffer because of regional extent
range_paed <- get_range(
occ_coord = obs_paed,
ecoreg = my_eco,
ecoreg_name = "EcoRegion",
res = 0.05,
  degrees_outlier = 0.5,
  buff_width_point = 0.5,
  buff_incrmt_pts_line = 0.5,
  buff_width_polygon = 0.5
)

# Plot
countries <- terra::vect(
  system.file("extdata", "world_countries.shp", package = "gbif.range")
)
terra::plot(terra::crop(countries,terra::ext(rst)), col = "#bcbddc")
terra::plot(
merge_range(range_paed),
add = TRUE,
col = "darkgreen",
axes = FALSE,
legend = FALSE
)
graphics::points(
obs_paed[, c("decimalLongitude","decimalLatitude")],
pch = 20,
col = "#99340470",
cex = 1
)

}
```

---

make_tiles *Create Tiled GBIF Geometry Queries*

Description

Divide a study extent into a set of smaller POLYGON() query strings that can be used with the GBIF geometry parameter.

Usage

```
make_tiles(geo, ntiles, sext = TRUE)
```

Arguments

geo	Spatial extent or geometry. Used to define the study area. Accepted classes are Extent, SpatExtent, SpatialPolygon, SpatialPolygonDataFrame, SpatVector, and sf. If NULL, the whole globe is used.
ntiles	Numeric. Approximate number of tiles to create.
sext	Logical. Should the corresponding SpatExtent objects also be returned?

Details

The input extent is converted into a regular grid of smaller query polygons. This is mainly intended to support tiled GBIF downloads when large extents would otherwise return too many records in a single query.

Value

If sext = TRUE, a list containing WKT POLYGON() strings and matching SpatExtent objects. Otherwise, a vector of WKT POLYGON() strings.

References

Chauvier, Y., Thuiller, W., Brun, P., Lavergne, S., Descombes, P., Karger, D. N., ... & Zimmermann, N. E. (2021). Influence of climate, soil, and land cover on plant species distribution in the European Alps. Ecological Monographs, 91(2), e01433. doi:10.1002/ecm.1433

See Also

get_gbif() which uses this tiling internally for large extents.

Examples

```
# Load the European Alps Extent
shp_path <- paste0(
system.file(package = "gbif.range"),
"/extdata/shp_lonlat.shp"
)
shp_lonlat <- terra::vect(shp_path)

# Apply the function to divide the extent in ~20 fragments
mt <- make_tiles(geo = shp_lonlat, ntiles = 20, sext = TRUE)
```

merge_range *Merge the polygons of a species range map*

Description

get_range() returns one polygon per occupied ecoregion. Those polygons do not overlap, so their areas are additive and the ecoregion origin of every piece stays visible. merge_range() dissolves them into a single polygon, or into groups given by by. The dissolve is kept out of get_range() because collapsing the ecoregions discards the ecoreg_name attribute. Rasterized output (format = "SpatRaster") never needs this function.

Usage

```
merge_range(x, by = NULL, format = c("SpatVector", "sf"))
```

Arguments

x	A getRange object returned by get_range(), or the SpatVector or sf object held in its rangeOutput field.
by	Character. Optional name of a field in x used to group the polygons: one feature is returned per distinct value. Note that on the default output of get_range() this is already the case, so by = ecoreg_name returns the input unchanged. It is useful for coarser groupings, such as a biome field, or for objects whose polygons were not built by get_range(). When NULL (default), every polygon is dissolved into a single feature.
format	Character. Output format. One of "SpatVector" (default) or "sf".

Value

A SpatVector or sf object holding the dissolved range. With by supplied, the grouping field is retained along with any other field that is constant within each group. With by = NULL the result carries no attributes, since no attribute of the original polygons describes the single merged geometry.

See Also

get_range() to build the range map.

Examples

```
# Load available ecoregions
eco_terra <- read_ecoreg(ecoreg_name = "eco_terra", save_dir = tempdir())

# Worldwide observations of the giant panda from GBIF
obs_am <- get_gbif(sp_name = "Ailuropoda melanoleuca")

# Guard against an unavailable remote source
if (gbif_have(eco_terra, obs_am)) {
```

```
# Range map: one polygon per occupied ecoregion
rng <- get_range(occ_coord = obs_am, ecoreg = eco_terra,
                 ecoreg_name = "ECO_NAME")
nrow(rng$rangeOutput)
sum(terra::expanse(rng$rangeOutput, unit = "km"))

# Dissolve into a single polygon. The area is unchanged, which shows the
# ecoregion polygons did not overlap.
merged <- merge_range(rng)
nrow(merged)
sum(terra::expanse(merged, unit = "km"))

}
```

`obs_filter` *Filter GBIF Records by Grid Cell*

Description

Reduce the spatial density of a `getGBIF` object by retaining a single record per grid cell and, optionally, removing cells with too few records.

Usage

```
obs_filter(gbifs, grid, threshold = NULL)
```

Arguments

`gbifs`	`getGBIF` object containing one or more species.
`grid`	Raster. Defining the target spatial resolution and extent. Accepted classes are `SpatRaster`, `RasterLayer`, `RasterBrick`, and `RasterStack`.
`threshold`	Optional integer. Specifying the minimum number of records a cell must contain to be retained.

Details

The function first collapses each species to one occurrence per grid cell. If `threshold` is supplied, cells with fewer than that many original records are then discarded.

Value

A data frame with the columns `Species`, `x`, and `y`, representing the filtered coordinates.

See Also

`get_gbif`() to produce the `getGBIF` object filtered by this function.

Examples

```
# Load data
shp_path <- paste0(
system.file(package = "gbif.range"),
"/extdata/shp_lonlat.shp"
)
shp_lonlat <- terra::vect(shp_path)
rst_path <- paste0(
system.file(package = "gbif.range"),
"/extdata/rst_enl.tif"
)
rst <- terra::rast(rst_path)

# Download observations for two plant species in the European Alps
obs_paed <- get_gbif(
sp_name = "Paederota bonarota",
geo = shp_lonlat
)
obs_saxi <- get_gbif(
sp_name = "Saxifraga cernua",
geo = shp_lonlat
)

# Guard against an unavailable remote source
if (gbif_have(obs_paed, obs_saxi)) {

# Test plot
terra::plot(shp_lonlat)
graphics::points(
obs_paed[, c("decimalLongitude","decimalLatitude")],
pch = 20,
col = "#238b4550",
cex = 1
)
graphics::points(
obs_saxi[, c("decimalLongitude","decimalLatitude")],
pch = 20,
col = "#99000d50",
cex = 1
)

# Combine both datasets
both_sp <- rbind(obs_paed, obs_saxi)

# Run function
obs_filt <- obs_filter(gbifs = both_sp, grid = rst, threshold = 4)

# Check new points
terra::plot(shp_lonlat)
graphics::points(
obs_filt[obs_filt$Species%in%"Paederota bonarota", c("x","y")],
pch = 20,
```

```
col = "#238b4550",
cex = 1
)
graphics::points(
obs_filt[obs_filt$Species%in%"Saxifraga cernua", c("x","y")],
pch = 20,
col = "#99000d50",
cex = 1
)

}
```

---

plot.gbifPackedSpatRaster

Plot a Packed Raster Range Saved by `species_csvs_to_ranges()`

Description

Plot a Packed Raster Range Saved by `species_csvs_to_ranges()`

Usage

```
## S3 method for class 'gbifPackedSpatRaster'
plot(x, ...)
```

Arguments

`x`	Object of class `gbifPackedSpatRaster`.
`...`	Additional arguments passed to `terra::plot()`.

Value

`x`, invisibly. Called for its side effect of plotting.

---

plot.gbifPackedSpatVector

Plot a Packed Vector Range Saved by `species_csvs_to_ranges()`

Description

Plot a Packed Vector Range Saved by `species_csvs_to_ranges()`

Usage

```
## S3 method for class 'gbifPackedSpatVector'
plot(x, ...)
```

Arguments

x	Object of class gbifPackedSpatVector.
...	Additional arguments passed to terra::plot().

Value

x, invisibly. Called for its side effect of plotting.

---

read_ecoreg *Read an Ecoregion Layer*

Description

Load one of the packaged ecoregion datasets listed in ecoreg_list(). If the requested data are not available locally, they are downloaded first.

Usage

```
read_ecoreg(ecoreg_name = "eco_terra", save_dir = NULL, format = "SpatVector")
```

Arguments

ecoreg_name	Character. File name of the ecoregion dataset to load. See ecoreg_list for available options.
save_dir	Character. Directory where the downloaded files are stored. Defaults to tempdir() unless the gbif.range.save_dir option or GBIF_RANGE_SAVE_DIR environment variable is set, in which case that location is used instead.
format	"SpatVector" (default) or "sf".

Details

Four datasets are currently available:

(1) eco_terra for terrestrial species, based on The Nature Conservancy version adapted from Olson et al. (2001).

(2) eco_marine for marine species, based on Spalding et al. (2007, 2012).

(3) eco_hd_marine, a higher-resolution marine version.

(4) eco_fresh for freshwater species, based on Abell et al. (2008).

Value

An ecoregion layer as a SpatVector or sf object, depending on format.

References

Olson, D. M., Dinerstein, E., Wikramanayake, E. D., Burgess, N. D., Powell, G. V. N., Underwood, E. C., D'Amico, J. A., Itoua, I., Strand, H. E., Morrison, J. C., Loucks, C. J., Allnutt, T. F., Ricketts, T. H., Kura, Y., Lamoreux, J. F., Wettengel, W. W., Hedao, P., Kassem, K. R. 2001. Terrestrial ecoregions of the world: a new map of life on Earth. BioScience 51(11):933-938. doi:10.1641/00063568(2001)051[0933:TEOTWA]2.0.CO;2

The Nature Conservancy (2009). Global Ecoregions, Major Habitat Types, Biogeographical Realms and The Nature Conservancy Terrestrial Assessment Units. GIS layers developed by The Nature Conservancy with multiple partners, combined from Olson et al. (2001), Bailey 1995 and Wiken 1986. Cambridge (UK): The Nature Conservancy.

Spalding, M. D., Fox, H. E., Allen, G. R., Davidson, N., Ferdaña, Z. A., Finlayson, M., Halpern, B. S., Jorge, M. A., Lombana, A., Lourie, S. A., Martin, K. D., McManus, E., Molnar, J., Recchia, C. A., Robertson, J. (2007). Marine Ecoregions of the World: A Bioregionalization of Coastal and Shelf Areas. BioScience, 57(7), 573-583. doi:10.1641/B570707

Spalding, M. D., Agostini, V. N., Rice, J., & Grant, S. M. (2012). Pelagic provinces of the world: a biogeographic classification of the world's surface pelagic waters. Ocean & Coastal Management, 60, 19-30. doi:10.1016/j.ocecoaman.2011.12.016

The Nature Conservancy (2012). Marine Ecoregions and Pelagic Provinces of the World. GIS layers developed by The Nature Conservancy with multiple partners, combined from Spalding et al. (2007) and Spalding et al. (2012). Cambridge (UK): The Nature Conservancy.

Abell, R., Thieme, M. L., Revenga, C., Bryer, M., Kottelat, M., Bogutskaya, N., Coad, B., Mandrak, N., Contreras Balderas, S., Bussing, W., Stiassny, M. L. J., Skelton, P., Allen, G. R., Unmack, P., Naseka, A., Ng, R., Sindorf, N., Robertson, J., Armijo, E., Higgins, J. V., Heibel, T. J., Wikramanayake, E., Olson, D., Lopez, H. L., Reis, R. E., Lundberg, J. G., Sabaj Perez, M. H., Petry, P. (2008). Freshwater Ecoregions of the World: A New Map of Biogeographic Units for Freshwater Biodiversity Conservation. BioScience, 58(5), 403-414. doi:10.1641/B580507

See Also

`ecoreg_list` for the list of available datasets, `get_ecoreg()` to force a fresh download, and `check_and_get_ecoreg()` for the underlying download check.

Examples

```
shp_eco_terra <- read_ecoreg("eco_terra", save_dir = tempdir())
terra::plot(shp_eco_terra)
```

`read_range_rds` *Read a Range File Saved by* `species_csvs_to_ranges()`

Description

Convenience wrapper around `readRDS()` for range files saved with `range_save_as = "rds"`.

Usage

```
read_range_rds(file)
```

Arguments

file	Path to an .rds range file created by species_csvs_to_ranges().

Value

An object of class getRange, with fields init.args and rangeOutput, as returned by get_range(). Note that init.args$ecoreg is NULL: the ecoregion layer is not serialized, so it must be re-supplied before the object can be passed to cv_range().

See Also

species_csvs_to_ranges() which creates the files read by this function.

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  gbif_file <- system.file("extdata", "occ_example_2sps.csv", package = "gbif.range")
  split_dir <- file.path(tempdir(), "gbif_read_range_split")
  range_dir <- file.path(tempdir(), "gbif_read_range_out")

  unlink(split_dir, recursive = TRUE)
  unlink(range_dir, recursive = TRUE)

  split_gbif_by_species(
    input_file = gbif_file,
    outdir = split_dir,
    chunk_size = 10,
    sep_in = "\t",
    sep_out = "\t",
    overwrite = TRUE,
    verbose = FALSE
  )

  occ_example <- utils::read.delim(gbif_file, sep = "\t", stringsAsFactors = FALSE)
  eco_terra <- read_ecoreg(
    "eco_terra",
    save_dir = tempdir()
  )

  # read_ecoreg() returns NULL if the remote source is unavailable
  if (gbif_have(eco_terra)) {

  eco_crop <- terra::crop(
    eco_terra,
    terra::ext(
      min(occ_example$decimalLongitude) - 5,
      max(occ_example$decimalLongitude) + 5,
      min(occ_example$decimalLatitude) - 5,
```

```
    max(occ_example$decimalLatitude) + 5
  )
)

range_summary <- species_csvs_to_ranges(
  species_dir = split_dir,
  ecoreg = eco_crop,
  ecoreg_name = "ECO_NAME",
  outdir = range_dir,
  range_save_as = "rds",
  sep_in = "\t",
  overwrite = TRUE,
  degrees_outlier = 30,
  clust_pts_outlier = 2,
  buff_width_point = 1,
  buff_incrmt_pts_line = 0.1,
  buff_width_polygon = 1,
  format = "SpatVector",
  verbose = FALSE
)

rg <- read_range_rds(range_summary$range_file[1])
class(rg)
nrow(rg$rangeOutput)
terra::plot(rg$rangeOutput)

}
}
```

`species_csvs_to_ranges` *Build Range Maps from Per-Species GBIF Files Saved on Disk*

Description

Read one occurrence file per species, prepare the minimal coordinate table needed by `get_range()`, and save one range output per species.

Usage

```
species_csvs_to_ranges(
  species_dir,
  ecoreg,
  ecoreg_name = NULL,
  outdir = file.path(tempdir(), "gbif_ranges"),
  occ_outdir = NULL,
  occ_save_as = c("none", "tsv", "rds"),
  range_save_as = c("rds", "gpkg", "tif"),
```

```
  deduplicate = TRUE,
  sep_in = "\t",
  overwrite = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

species_dir	Character. Directory containing files created by split_gbif_by_species().
ecoreg	Spatial ecoregion layer in WGS84. Accepted classes are SpatialPolygonsDataFrame, SpatVector, and sf. This can be a downloaded layer from read_ecoreg() or a custom layer created by make_ecoreg().
ecoreg_name	Character. String naming the field in ecoreg that defines ecoregion categories. For eco_terra, for example, "ECO_NAME" is the most detailed level. When ecoreg comes from make_ecoreg(), "EcoRegion" is used automatically.
outdir	Character. Directory where range files will be saved.
occ_outdir	Optional character. Directory where the minimal occurrence tables passed to get_range() will also be saved.
occ_save_as	"none", "tsv", or "rds". File format for minimal occurrence tables.
range_save_as	"rds", "gpkg", or "tif". File format for range outputs.
deduplicate	Logical. Should identical longitude-latitude pairs be collapsed before range inference?
sep_in	Character. Field separator used by the per-species occurrence files.
overwrite	Logical. If TRUE, existing outputs in outdir/occ_outdir are replaced.
verbose	Logical. Should progress messages be printed?
...	Additional arguments passed to get_range().

Details

The function is intended to work seamlessly with split_gbif_by_species(). It reads each species file sequentially, adds the single-species input_search column expected by get_range(), and keeps only the coordinate columns needed for range construction.

Value

A data frame summarizing the processed files, with one row per range and the columns species_key, species_name, n_points, occ_file, and range_file.

See Also

split_gbif_by_species() and read_range_rds().

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  gbif_file <- system.file("extdata", "occ_example_2sps.csv", package = "gbif.range")
  split_dir <- file.path(tempdir(), "gbif_species_help")
  occ_dir <- file.path(tempdir(), "gbif_occ_help")
  range_dir <- file.path(tempdir(), "gbif_range_help")

  unlink(split_dir, recursive = TRUE)
  unlink(occ_dir, recursive = TRUE)
  unlink(range_dir, recursive = TRUE)

  split_gbif_by_species(
    input_file = gbif_file,
    outdir = split_dir,
    chunk_size = 10,
    sep_in = "\t",
    sep_out = "\t",
    overwrite = TRUE,
    verbose = FALSE
  )

  # Crop the packaged terrestrial ecoregions to the extent of the example
  # occurrences so the help example stays lightweight.
  occ_example <- utils::read.delim(gbif_file, sep = "\t", stringsAsFactors = FALSE)
  eco_terra <- read_ecoreg(
    "eco_terra",
    save_dir = tempdir()
    )

  # read_ecoreg() returns NULL if the remote source is unavailable
  if (gbif_have(eco_terra)) {

  eco_crop <- terra::crop(
    eco_terra,
    terra::ext(
      min(occ_example$decimalLongitude) - 5,
      max(occ_example$decimalLongitude) + 5,
      min(occ_example$decimalLatitude) - 5,
      max(occ_example$decimalLatitude) + 5
    )
  )

  range_summary <- species_csvs_to_ranges(
    species_dir = split_dir,
    ecoreg = eco_crop,
    ecoreg_name = "ECO_NAME",
    outdir = range_dir,
    occ_outdir = occ_dir,
    occ_save_as = "tsv",
    range_save_as = "rds",
    sep_in = "\t",
    overwrite = TRUE,
```

```
    degrees_outlier = 30,
    clust_pts_outlier = 2,
    buff_width_point = 1,
    buff_incrmt_pts_line = 0.1,
    buff_width_polygon = 1,
    format = "SpatVector",
    verbose = FALSE
  )

  range_summary[, c("species_name", "n_points", "range_file")]

  }
}
```

`split_gbif_by_species` *Split a Downloaded GBIF Table into One File per Species*

Description

Stream a large GBIF export from disk in chunks and write one occurrence file per species or GBIF taxon key. The function is designed for multi-species tables that are too large to load fully into memory.

Usage

```
split_gbif_by_species(
  input_file,
  outdir = file.path(tempdir(), "gbif_by_species"),
  chunk_size = 1e+05,
  select_cols = c("speciesKey", "species", "scientificName", "decimalLongitude",
    "decimalLatitude"),
  sep_in = "\t",
  sep_out = "\t",
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`input_file`	Character. Path to a tabular GBIF export already stored on disk.
`outdir`	Character. Directory where one per-species file will be written.
`chunk_size`	Integer. Number of rows read at a time. Larger values are usually faster, whereas smaller values reduce peak memory use.
`select_cols`	Character. Vector of columns to keep from the original file. The defaults retain the taxon key, species labels, and geographic coordinates needed by downstream range workflows.

sep_in	Character. Field separator Used by the input file. GBIF downloads are usually tab-delimited.
sep_out	Character. Field separator used for the saved species files.
overwrite	Logical. If TRUE, existing batch files created by this function in outdir are removed before writing new ones.
verbose	Logical. Should progress messages be printed?

Details

Output file names follow the pattern occurrences_speciesKey_<key>_<species>.csv. The extension is kept as .csv for convenience, even when the file remains tab-delimited.

Value

A data frame summarizing the written files, with one row per species key and the columns species_key, species_name, n_records, and species_file.

See Also

species_csvs_to_ranges() to process the written species files sequentially with get_range().

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  gbif_file <- system.file("extdata", "occ_example_2sps.csv", package = "gbif.range")
  split_dir <- file.path(tempdir(), "gbif_split_help")

  # Remove earlier temporary outputs so the example can be rerun cleanly.
  unlink(split_dir, recursive = TRUE)

  split_summary <- split_gbif_by_species(
    input_file = gbif_file,
    outdir = split_dir,
    chunk_size = 10,
    sep_in = "\t",
    sep_out = "\t",
    overwrite = TRUE,
    verbose = FALSE
  )

  split_summary[, c("species_name", "n_records", "species_file")]
}
```

Index