Package 'biofetchR'

August 19, 2026

Type Package

Title Download, Clean, Classify, Enrich and Export Biodiversity Occurrence Data

Version 0.1.1

Description Downloads, imports, cleans, classifies, enriches and exports biodiversity occurrence data, with an emphasis on reproducible Global Biodiversity Information Facility (GBIF) <https://api.gbif.org/v1/> workflows. The package supports batch occurrence downloads, taxonomic standardisation, coordinate cleaning, optional spatial thinning, spatial attribution and structured export of processed occurrence records and audit outputs. Terrestrial and freshwater workflows can join records to administrative units, protected areas, freshwater ecoregions, basins, rivers, lakes, reservoirs, wetlands and other contextual spatial overlays. Marine workflows support offshore and coastal records through joins to Marine Regions <https://www.marineregions.org/> style layers, Exclusive Economic Zone (EEZ) units, marine ecoregions, Large Marine Ecosystems and user-supplied marine overlays. The package also supports native-range and invasive-status evidence workflows using the World Register of Marine Species (WoRMS) <https://www.marinespecies.org/>, evidence derived from Standardising and Integrating Alien Species (SInAS) <https://zenodo.org/records/18220953>, and Global Register of Introduced and Invasive Species (GRIIS) <https://griis.org/> style species-country records. These tools are intended for biodiversity, macroecological and invasion-biology analyses where occurrence records need to be processed consistently, transparently and reproducibly.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports cli, CoordinateCleaner, countrycode, curl, dplyr, geodata, jsonlite, mregions2, readr, rgbif, rlang, sf, tibble

Suggests geosphere, ggplot2, httr, knitr, lwgeom, mapme.biodiversity, osmdata, readxl, remotes, rmarkdown, rnaturalearth, stringi, stringr, terra, testthat (>= 3.0.0), tidyr, worrms

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Darren Stuart [aut, cre]

Maintainer Darren Stuart <dstuart04@qub.ac.uk>

Repository CRAN

Date/Publication 2026-08-19 19:50:02 UTC

Contents

`append_summary_row` *Append a row to a GBIF processing summary table*

Description

Record the processing outcome for one species x region combination and append it to a summary table created by `initialize_summary()`.

Usage

```
append_summary_row(
  summary_tbl,
  species,
  region_id,
  region_type,
  n_total,
  n_cleaned,
  n_thinned,
  output_file,
  status,
  quiet = FALSE
)
```

Arguments

`summary_tbl`	A data frame or tibble, usually returned by `initialize_summary()`.
`species`	Character scalar. Scientific name or accepted taxon label.
`region_id`	Character scalar. Spatial recipient identifier, such as a GADM code, EEZ name or marine-region identifier.
`region_type`	Character scalar. Spatial-recipient family or overlay label, such as `"GADM"` or `"EEZ"`.
`n_total`	Integer-compatible value. Number of records before cleaning.
`n_cleaned`	Integer-compatible value. Number of records after coordinate cleaning.
`n_thinned`	Integer-compatible value. Number of records after spatial thinning.
`output_file`	Character scalar. Exported CSV path, or `NA` when output is retained in memory.
`status`	Character scalar. Processing status label.
`quiet`	Logical. If `TRUE`, suppress the console status message.

Details

This helper performs light type coercion for the count fields so that summary tables remain stable when upstream pipeline steps return numeric, integer or missing values. It does not validate that `n_cleaned <= n_total` or `n_thinned <= n_cleaned`, because some workflows may report values from different stages or use specialised cleaning/thinning backends.

When `quiet = FALSE`, the helper prints a short success message using `cli::cli_alert_success()`. Set `quiet = TRUE` in batch tests, package checks or non-interactive workflows where console output should be suppressed.

Value

A tibble containing the existing rows in `summary_tbl` plus one appended processing-summary row. The appended row records the supplied `species`, `region_id`, `region_type`, record counts, `output_file` and `status`. Count fields are coerced to integer so the summary schema remains stable across pipeline steps. The returned table is used to accumulate one row per processed species x spatial-recipient combination in terrestrial/freshwater and marine GBIF workflows.

See Also

Other processing summary helpers: `initialize_summary()`

Examples

```
summary_tbl <- initialize_summary()

summary_tbl <- append_summary_row(
  summary_tbl = summary_tbl,
  species = "Rattus norvegicus",
  region_id = "GB",
  region_type = "GADM",
  n_total = 100,
  n_cleaned = 90,
  n_thinned = 25,
  output_file = "Rattus_norvegicus__GB.csv",
  status = "success",
  quiet = TRUE
)

summary_tbl
```

`bf_apply_manual_taxonomy_fixes` *Apply manual taxonomy name and rank fixes to a data frame*

Description

Apply manual taxonomy name and rank fixes to a data frame

Usage

```
bf_apply_manual_taxonomy_fixes(
  x,
  manual_fixes,
  name_col = "species",
  overwrite_names = TRUE,
  overwrite_ranks = FALSE,
 rank_cols = c("kingdom", "phylum", "class", "order", "family", "genus", "species_rank")
)
```

Arguments

x	Data frame.
manual_fixes	Named character vector or data frame with manual fixes.
name_col	Name column in x.
overwrite_names	Replace names when a manual fix is available.
overwrite_ranks	Overwrite existing rank columns when manual rank values are supplied.
rank_cols	Rank columns to fill if they exist in both tables.

Value

A data frame with the same rows as x and with manual taxonomy fixes applied where matching keys are found. If overwrite_names = TRUE, values in name_col are replaced by matched manual replacement names. If overwrite_ranks = TRUE, matching rank columns are also updated from manual_fixes; otherwise rank values are filled only where the input value is missing or blank. If no usable manual fixes are supplied, the original input data frame is returned unchanged.

---

bf_attach_gbif_taxonomy

Attach GBIF taxonomy to a data frame

Description

Attach GBIF taxonomy to a data frame

Usage

```
bf_attach_gbif_taxonomy(x, name_col = "species", prefix = "tax_", ...)
```

Arguments

x	Data frame.
name_col	Column containing taxon names.
prefix	Prefix for attached taxonomy columns. Use "" for no prefix.
...	Passed to bf_resolve_gbif_taxonomy_batch().

Value

A data frame with the same rows as x and GBIF taxonomy-resolution columns joined by name_col. The attached columns are produced by bf_resolve_gbif_taxonomy_batch() and, by default, are prefixed with prefix. They include resolved names, GBIF usage keys, rank, classification fields, match diagnostics and resolver provenance. The output is used to keep original records together with their taxonomy audit information.

`bf_attach_griis_status` *Attach GRIIS status to a species or species-country table*

Description

Adds GRIIS listing and invasive-status evidence to an input table. The function can join by species + country for terrestrial/freshwater workflows or by species only for marine/global workflows.

Usage

```
bf_attach_griis_status(
  df,
  species_col = "species",
  iso2c_col = NULL,
  iso3c_col = NULL,
  country_col = NULL,
  griis = NULL,
  cache_dir = NULL,
  force_refresh = FALSE,
  require_country = TRUE,
  quiet = FALSE
)
```

Arguments

df	Input data frame.
species_col	Name of the column containing species names.
iso2c_col	Optional name of an ISO2 recipient-country column.
iso3c_col	Optional name of an ISO3 recipient-country column.
country_col	Optional name of a recipient country-name column.
griis	Optional raw or standardised GRIIS table. If NULL, GRIIS is read with bf_read_griis().
cache_dir	Cache directory used when griis = NULL. Must be supplied explicitly when the helper needs to download/read GRIIS internally. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; if TRUE, re-download/re-read GRIIS when the helper loads it.

require_country
: Logical. If TRUE, join by species + recipient country. If FALSE, join by species only.

quiet
: Logical; if TRUE, suppress progress messages.

Details

When require_country = TRUE, at least one of iso2c_col, iso3c_col or country_col must identify the recipient country. These fields are resolved to ISO3 and joined against a species x country GRIIS lookup.

When require_country = FALSE, no country parsing is attempted and the join is species-only. This is important for marine pipelines where the input table may contain only species names before marine spatial overlays are assigned.

The function appends:

- griis_listed: whether the species/country or species-only record appears in GRIIS;
- griis_invasive: whether any matching GRIIS row is flagged invasive;
- griis_status: a compact categorical summary of the match.

Value

A tibble containing the input rows with GRIIS evidence columns appended. The returned object preserves the input columns and adds fields such as griis_listed, griis_invasive, griis_status, griis_any_invasive_field_present, griis_n_matches and supporting matched-name/status metadata where available. griis_status is a character summary with values such as "listed_invasive", "listed_introduced_or_alien" and "not_listed_or_no_match". These columns provide GRIIS evidence for filtering and auditing; non-matches should not be interpreted as confirmed native status, absence or lack of impact.

Interpretation

GRIIS columns appended by this helper are evidence fields for filtering and audit. A row with griis_listed = FALSE means no matching GRIIS evidence was found under the selected matching mode; it should not be interpreted as proof that the species is native, absent or harmless in that recipient region.

See Also

Other GRIIS origin-evidence helpers: bf_download_griis(), bf_filter_griis_invasive(), bf_find_griis_table(), bf_griis_lookup(), bf_read_griis(), bf_standardise_griis(), bf_unpack_griis()

Examples

```
raw <- data.frame(
  species = "Example species",
  scientificName = "Example species",
  countryCode_alpha2 = "GB",
  countryCode_alpha3 = "GBR",
```

```
  isInvasive = "yes",
  stringsAsFactors = FALSE
)
input <- data.frame(species = "Example species", iso2c = "GB")
bf_attach_griis_status(input, iso2c_col = "iso2c", griis = raw)
```

`bf_attach_native_status` *Attach native/non-native recipient status to a data frame*

Description

Joins species-level native-origin evidence to an input table and, when a recipient country is available, classifies each row as native, non-native or origin-unknown for that recipient.

Usage

```
bf_attach_native_status(
  df,
  species_col = "species",
  iso2c_col = NULL,
  iso3c_col = NULL,
  country_col = NULL,
  native_ranges = NULL,
  native_species_col = "species",
  origin_iso3_col = NULL,
  require_country = TRUE,
  strategy = c("tiered", "union"),
  native_filter_mode = c("audit_only", "non_native_only", "keep_non_native",
   "non_native_or_unknown", "keep_non_native_or_unknown", "native_only", "keep_native"),
  max_origins = 100L,
  use_native_web = FALSE,
  native_web_sources = c("sinas", "gbif", "worms"),
  native_web_cache_dir = NULL,
  native_web_force_refresh = FALSE,
  native_web_sleep_sec = 0.25,
  native_web_quiet = quiet,
  native_web_sinas_main_path = NULL,
  native_web_sinas_alllocations_path = NULL,
  native_web_sinas_fulltaxa_path = NULL,
  export_native_web_audit = FALSE,
  native_web_audit_dir = NULL,
  native_web_audit_prefix = "native_web",
  quiet = FALSE
)
```

Arguments

df	Input data frame.
species_col	Species column in df.
iso2c_col	Optional ISO2 recipient column in df.
iso3c_col	Optional ISO3 recipient column in df.
country_col	Optional recipient country-name column in df. Used only when require_country = TRUE and ISO2/ISO3 columns are unavailable.
native_ranges	Native-origin evidence table.
native_species_col	Species column in native_ranges.
origin_iso3_col	Optional origin ISO3 column in native_ranges.
require_country	Logical; when TRUE, classify recipient status using recipient ISO2/ISO3. When FALSE, attach species-level origin availability.
strategy	Native-origin column strategy passed to bf_native_range_lookup().
native_filter_mode	One of "audit_only", "non_native_only", "keep_non_native", "non_native_or_unknown", "keep_non_native_or_unknown", "native_only" or "keep_native".
max_origins	Maximum origin count per species.
use_native_web	Logical. If TRUE and native_ranges = NULL, compile native-origin evidence from provider helpers before classification.
native_web_sources	Character vector of provider names passed to bf_fetch_native_ranges_web(), commonly c("sinas", "gbif", "worms").
native_web_cache_dir	Cache directory for web/API native-origin evidence. Must be supplied explicitly when use_native_web = TRUE. In examples, tests and vignettes, use a path under tempdir().
native_web_force_refresh	Logical; refresh cached provider evidence where supported.
native_web_sleep_sec	Delay in seconds between provider requests where supported.
native_web_quiet	Logical; suppress provider-specific progress messages.
native_web_sinas_main_path, native_web_sinas_alllocations_path, native_web_sinas_fulltaxa_path	Optional local SInAS resource paths passed through to the SInAS provider.
export_native_web_audit	Logical; write native-web evidence audit files when provider compilation is used and the writer helper is available.
native_web_audit_dir	Directory for optional native-web audit files. If NULL, audit files are written to native_web_cache_dir when export_native_web_audit = TRUE.

native_web_audit_prefix
: File prefix for optional native-web audit files.

quiet
: Logical; suppress messages.

Details

This is the core classifier used by the terrestrial/freshwater and marine pipelines. With `require_country = TRUE`, the helper resolves recipient country information to ISO3 and compares it with each species' native-origin ISO3 set. With `require_country = FALSE`, it attaches species-level origin availability without making recipient-level native/non-native claims.

If `native_ranges = NULL` and `use_native_web = TRUE`, provider evidence is compiled through `bf_fetch_native_ranges_web()` before classification. This allows SInAS/GBIF/WoRMS evidence to be generated inside the helper, while still keeping the classification logic centralised here.

Value

A tibble containing the input rows with native-origin and recipient-status columns attached. The returned object preserves the input columns and adds fields such as `native_name_key`, `native_name`, `native_origin_iso3`, `native_has_origin`, `native_sources_used`, `native_recipient_iso3`, `native_is_native_recipient`, `native_is_non_native_recipient`, `native_status`, `native_filter_keep` and `native_filter_rejection_reason`. When `require_country = TRUE`, `native_status` classifies each row as `"native_recipient"`, `"non_native_recipient"` or `"origin_unknown"` by comparing the recipient ISO3 code with the species-level native-origin ISO3 set. When `require_country = FALSE`, the function attaches species-level origin availability without making recipient-level native/non-native claims. Strict `*_only` filter modes return only retained rows; `keep_*` modes annotate retention decisions without dropping rows.

Interpretation

`origin_unknown` means that the available native-origin evidence did not confirm a usable origin set for that species. It should not be interpreted as proof that the species is native, non-native, absent or data-deficient in an ecological sense.

Data source and attribution

This function classifies evidence supplied by other sources. If evidence is generated from SInAS, GBIF, WoRMS or another provider, cite the exact source, version/release and access date used to create `native_ranges`.

See Also

Other native-origin status helpers: `bf_filter_native()`, `bf_filter_non_native()`, `bf_native_range_lookup()`, `bf_native_status_summary()`, `bf_reconcile_griis_native_status()`, `bf_standardise_native_ranges()`

`bf_available_marine_overlays` *List package-supported marine overlays*

Description

Returns the marine spatial overlays that biofetchR can load through its package-managed Marine Regions workflow.

Usage

```
bf_available_marine_overlays(include_aliases = FALSE, include_extended = FALSE)
```

Arguments

`include_aliases`
: Logical. If `TRUE`, include a list-column containing accepted aliases and candidate Marine Regions product identifiers.

`include_extended`
: Logical. If `TRUE`, include all registered overlays. If `FALSE`, return only the core overlays.

Details

By default, this helper returns the core overlays that are expected to be resolvable in most mregions2/Marine Regions installations: `eez`, `meow`, `lme` and `iho`. Set `include_extended = TRUE` to list the full registry of supported extended overlays used by the marine pipeline and release-gate checks.

The returned overlay names can be supplied to `process_gbif_marine_pipeline()` through its marine overlay argument. Aliases can also be inspected with `include_aliases = TRUE`.

Value

A data frame listing marine overlays registered by biofetchR. The returned object contains `overlay`, a character canonical overlay name; `description`, a character description of the spatial unit type; and `workflow`, a character label identifying the overlay as part of the marine workflow. When `include_aliases = TRUE`, the returned data frame also contains `aliases`, a list-column of accepted aliases and candidate Marine Regions product identifiers. The table is used to inspect valid overlay names before loading or passing them to marine processing workflows.

Scope

This helper only lists overlay names registered by biofetchR. It does not download or cache Marine Regions products. Data access happens in `bf_load_marine_regions_overlay()`.

See Also

Other marine overlay loaders: `bf_load_marine_regions_overlay()`, `bf_marine_overlay_canonical()`

Examples

```
bf_available_marine_overlays()
bf_available_marine_overlays(include_extended = TRUE)
```

`bf_available_native_web_sources` *List web sources supported for native-range evidence*

Description

Returns the web/API sources currently implemented by `bf_fetch_native_ranges_web()`. These are optional enrichment sources used to compile species-level native-origin evidence when a complete local native range table is not available.

Usage

```
bf_available_native_web_sources()
```

Value

A data frame listing native-origin web/API evidence sources supported by biofetchR. The returned table contains `source`, the source identifier accepted by `bf_fetch_native_ranges_web()`; `description`, a short explanation of the evidence provider; `access_type`, the way the source is accessed or cached; and `default`, a logical flag indicating whether the source is included in the default source set. The table is intended for inspecting valid source names before requesting web-derived native-origin evidence.

Relationship to other helpers

This function lists optional evidence providers only. The returned sources are consumed by `bf_fetch_native_ranges_web(` and then passed to the core native-status classifier in `bf_attach_native_status()`.

See Also

Other native-range web-source helpers: `bf_fetch_native_ranges_web()`, `bf_web_native_gbif()`, `bf_web_native_worms()`, `bf_write_native_web_outputs()`

Examples

```
bf_available_native_web_sources()
```

`bf_bind_gbif_chunks` *Harmonise and bind GBIF result chunks*

Description

Bind a list of GBIF result chunks into a single spatial object while standardising geometry handling and reducing mixed-column type conflicts.

Usage

```
bf_bind_gbif_chunks(x_list, coerce_all_to_character = TRUE)
```

Arguments

`x_list`	List of GBIF result chunks. Each element should be an `sf` object or a data frame containing `decimalLongitude` and `decimalLatitude` columns. Empty, `NULL` or unsupported chunks are silently dropped.
`coerce_all_to_character`	Logical. If `TRUE`, all non-geometry columns are coerced to character before binding to avoid type conflicts across GBIF chunks. If `FALSE`, existing column classes are retained where possible.

Details

GBIF imports can produce chunks with heterogeneous columns, missing fields or incompatible column classes across species, countries, downloads or publishing datasets. `bf_bind_gbif_chunks()` keeps only non-empty chunks, converts plain data frames with `decimalLongitude` and `decimalLatitude` columns into `sf` point objects, standardises the geometry column name to `"geometry"`, and transforms spatial inputs to WGS84 when needed.

The helper then aligns the union of all non-geometry columns across valid chunks and, by default, coerces those attributes to character before binding. This conservative coercion avoids common row-binding failures caused by GBIF fields switching between integer, numeric, logical, factor and character classes across chunks.

The function intentionally avoids relying on `sf::st_geometry_name()` so that it remains compatible with older `sf` versions used on some Windows or HPC installations.

Value

An `sf` point object when at least one valid GBIF chunk is supplied. The returned object contains the union of non-geometry columns across valid chunks, a standardised `geometry` column, and WGS84 point geometries derived from existing `sf` geometries or from `decimalLongitude` and `decimalLatitude` columns. When `coerce_all_to_character = TRUE`, all non-geometry columns are coerced to character before binding to avoid mixed type failures across GBIF chunks. If no valid chunks are available, the function returns an empty tibble.

Relationship to older helpers

This function replaces the older, ambiguous `harmonize_column_types()` helper for GBIF result chunks. It is named explicitly to show that it returns one bound object, not a harmonised list.

Examples

```
x1 <- data.frame(
  species = "Example species",
  decimalLongitude = -6.26,
  decimalLatitude = 53.35,
  individualCount = 1
)

x2 <- data.frame(
  species = "Example species",
  decimalLongitude = -6.30,
  decimalLatitude = 53.40,
  basisOfRecord = "HUMAN_OBSERVATION"
)

if (requireNamespace("sf", quietly = TRUE)) {
  bf_bind_gbif_chunks(list(x1, x2))
}
```

---

bf_cache_dir *Get a writable cache directory for biofetchR spatial layers*

Description

This helper returns a cache directory (creating it if needed) where biofetchR can store downloaded spatial layers (zips + extracted shapefiles). Users are not expected to download or manage raw data files themselves.

Usage

```
bf_cache_dir(cache_dir = NULL, subdir = NULL)
```

Arguments

`cache_dir`	Character. Root cache directory. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`subdir`	Optional character. Subdirectory to create/use inside cache_dir.

Value

A character vector of length one giving the normalised path to a writable cache directory. The directory is created if it does not already exist, so the function is also called for the side effect of preparing a cache location for downloaded spatial layers. If `subdir` is supplied, the returned path points to that subdirectory inside the cache root.

`bf_check_taxonomic_resolution` *Check taxonomic resolution quality*

Description

Check taxonomic resolution quality

Usage

```
bf_check_taxonomic_resolution(taxonomy_tbl, min_confidence = 90)
```

Arguments

`taxonomy_tbl`	Output from `bf_resolve_gbif_taxonomy_batch()` or a data frame with equivalent columns.
`min_confidence`	Optional minimum GBIF confidence threshold.

Value

A tibble containing the subset of rows from `taxonomy_tbl` with potential taxonomy-resolution issues. The output preserves the available taxonomy columns and adds an `issue` column describing the detected problem, such as `"non_taxon_skipped"`, `"unresolved"`, `"missing_gbif_taxon_key"`, `"low_confidence"`, `"non_exact_match"` or `"higher_rank_match"`. An empty tibble indicates that no rows were flagged by the selected checks.

`bf_clean_taxon_names` *Clean taxon names before taxonomy resolution*

Description

Applies conservative string cleaning before GBIF or WoRMS lookup. The helper removes simple authorship fragments, parenthetical notes, selected descriptor words and implausible non-taxonomic strings while retaining plausible genus, binomial and trinomial names for downstream auditing.

Usage

```
bf_clean_taxon_names(x, max_tokens = 3, keep_genus_for_open_names = TRUE)
```

Arguments

x	Character vector of raw taxon names.
max_tokens	Integer maximum number of whitespace-separated tokens to retain after cleaning. The default, 3, preserves trinomial names.
keep_genus_for_open_names	Logical. If TRUE, strings such as Genus sp. or Genus spp. are converted to Genus; if FALSE, they are converted to NA_character_.

Value

A character vector with the same length as x. Each element contains a cleaned taxon-name candidate after whitespace, authority, punctuation, descriptor and open-name handling. Blank, non-taxonomic or implausible entries are returned as NA_character_. The returned names are intended for GBIF or WoRMS taxonomy resolution and for audit columns in downstream pipelines.

---

bf_download_griis *Download the GRIIS country compendium*

Description

Downloads the country-level Global Register of Introduced and Invasive Species (GRIIS) compendium and stores it in a local cache. The downloader is designed to be robust on Windows: files are first downloaded to a .tmp path and then promoted to the final cache path by rename or copy.

Usage

```
bf_download_griis(
  cache_dir = NULL,
  force_refresh = FALSE,
  source_url = .bf_griis_default_source_url(),
  quiet = FALSE
)
```

Arguments

cache_dir	Directory used for the cached GRIIS file. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; if TRUE, re-download even if a cached file is already present.
source_url	Character URL for the country-level GRIIS compendium. The default points to the Zenodo-hosted country compendium CSV used by biofetchR.
quiet	Logical; if TRUE, suppress progress messages.

Details

biofetchR requires the country-level GRIIS table because the terrestrial and freshwater workflows use species x recipient-country evidence. The helper therefore rejects the `griis_meta_recoded` metadata/DwC-A archive, which lacks the required country-level fields used by the GRIIS gate.

If a previous run left a valid temporary file in the cache, the helper attempts to finalise that file before downloading again. This avoids unnecessary repeated downloads when a prior session succeeded but file finalisation failed.

Value

A character vector of length one giving the normalised path to the cached GRIIS country compendium file, using forward slashes. The path points to either an existing valid cached file or a newly downloaded file. The function is also called for the side effect of downloading, validating and caching the external GRIIS source file.

Data source and attribution

This function downloads the external GRIIS country compendium into a local user cache. biofetchR does not redistribute or claim ownership of the source data. Users should cite the GRIIS data source, version/record and associated provider guidance used in their analysis.

See Also

Other GRIIS origin-evidence helpers: `bf_attach_griis_status()`, `bf_filter_griis_invasive()`, `bf_find_griis_table()`, `bf_griis_lookup()`, `bf_read_griis()`, `bf_standardise_griis()`, `bf_unpack_griis()`

Examples

```
if (interactive()) {
  path <- bf_download_griis(
    cache_dir = file.path(tempdir(), "biofetchR_griis"),
    quiet = FALSE
  )

  file.exists(path)
}
```

`bf_download_sinas_resources`

Download and locate required SInAS resources

Description

Downloads, caches and resolves the SInAS 3.1.1 resources required by the biofetchR native-evidence workflow. Existing local files can be supplied to bypass downloading. Otherwise, the function downloads the main SInAS table, extracts `AllLocations.xlsx` from the configuration archive, and recovers `SInAS_3.1.1_FullTaxaList.csv` from the output archive for the default record.

Usage

```
bf_download_sinas_resources(
  cache_dir = NULL,
  force_refresh = FALSE,
  quiet = FALSE,
  main_path = NULL,
  allloc_path = NULL,
  fulltaxa_path = NULL,
  record_id = "18220953",
  main_url = bf_sinas_default_urls(record_id)[["main_csv"]],
  fulltaxa_url = NULL,
  config_zip_url = bf_sinas_default_urls(record_id)[["config_zip"]],
  output_zip_url = bf_sinas_default_urls(record_id)[["output_zip"]]
)
```

Arguments

`cache_dir`	Character scalar. Local cache directory for downloaded and extracted SInAS resources. Must be supplied explicitly when any required SInAS resource needs to be downloaded or extracted. If `main_path`, `allloc_path` and `fulltaxa_path` all point to existing local files, `cache_dir` is not required. In examples, tests and vignettes, use a path under `tempdir()`.
`force_refresh`	Logical scalar. If `TRUE`, re-download and re-extract cached resources even when existing files are present.
`quiet`	Logical scalar. If `TRUE`, suppress progress messages.
`main_path`	Optional character scalar. Existing local path to `SInAS_3.1.1.csv`. When supplied and valid, this path is used instead of downloading `main_url`.
`allloc_path`	Optional character scalar. Existing local path to `AllLocations.xlsx`. When supplied and valid, this path is used instead of searching the extracted configuration archive.
`fulltaxa_path`	Optional character scalar. Existing local path to `SInAS_3.1.1_FullTaxaList.csv`. When supplied and valid, this path is used instead of downloading or extracting FullTaxaList.
`record_id`	Character scalar. Zenodo record identifier used by `bf_sinas_default_urls()` to construct default URLs.
`main_url`	Character scalar. Direct URL for the main SInAS CSV.
`fulltaxa_url`	Optional character scalar. Direct URL for a standalone FullTaxaList CSV. Defaults to `NULL` because the default SInAS 3.1.1 record stores FullTaxaList inside the output ZIP rather than as a standalone file.

`config_zip_url` Character scalar. Direct URL for the configuration ZIP containing `AllLocations.xlsx`.

`output_zip_url` Character scalar. Direct URL for the output ZIP containing `SInAS_3.1.1_FullTaxaList.csv` for the default record.

Value

A named list of normalised local file paths to the SInAS resources required by biofetchR native-origin evidence workflows. The list always contains `main_csv`, the local path to `SInAS_3.1.1.csv`; `alllocations_xlsx`, the local path to `AllLocations.xlsx` or an equivalent location table found in the extracted configuration archive; `fulltaxa_csv`, the local path to `SInAS_3.1.1_FullTaxaList.csv` or a matching FullTaxaList file recovered from the output archive; `config_zip`, the cached configuration archive path; and `config_dir`, the extracted configuration directory path. When FullTaxaList is recovered from `All_Output_Files_SInAS_v3.1.1.zip`, the list also contains `output_zip` and `output_dir`. User-supplied local paths are returned in normalised form when provided and valid; otherwise paths point to files downloaded or extracted inside `cache_dir`.

Cache behaviour

Cached files are reused when present and above the minimum size threshold. Set `force_refresh = TRUE` to re-download archives and rebuild extracted directories. User-supplied local paths always take precedence over downloads.

Data access and attribution

biofetchR does not ship or redistribute SInAS resources. This function only downloads files into a user-managed local cache. Users should check the relevant SInAS/Zenodo record for licence, citation and attribution requirements before publishing, redistributing or archiving derived outputs.

See Also

Other SInAS resource helpers: `bf_sinas_default_urls()`

Examples

```
if (interactive()) {
  sinas_paths <- bf_download_sinas_resources(
    cache_dir = file.path(tempdir(), "biofetchR_sinas"),
    quiet = FALSE
  )

  names(sinas_paths)
  sinas_paths$main_csv
}
```

`bf_enrich_raster_context` *Enrich GBIF points with raster context layers*

Description

Adds raster-derived covariates to an `sf` point object. Supported contexts are currently WorldCover, SoilGrids and Human Footprint. Raster files are prepared through remote-first helper functions unless local sources are supplied.

Usage

```
bf_enrich_raster_context(
  sf_points,
  raster_context = character(0),
  cache_dir = NULL,
  worldcover_source = NULL,
  worldcover_year = 2020,
  worldcover_buffer_m = 0,
  soilgrids_sources = NULL,
  soilgrids_vars = NULL,
  soilgrids_buffer_m = 0,
  human_footprint_source = NULL,
  human_footprint_buffer_m = 0,
  quiet = TRUE
)
```

Arguments

`sf_points`	Point `sf` object. Empty inputs, non-`sf` inputs or calls with no requested `raster_context` are returned unchanged where applicable.
`raster_context`	Character vector of raster contexts to extract. Supported values are `"worldcover"`, `"soilgrids"` and `"human_footprint"`.
`cache_dir`	Character. Directory used for raster download and preparation caches. Must be supplied explicitly when raster context layers are requested. In examples, tests and vignettes, use a path under `tempdir()`.
`worldcover_source`	Optional local path or URL for a WorldCover raster. When `NULL`, the default WorldCover Zenodo record for `worldcover_year` is used.
`worldcover_year`	Numeric or character WorldCover year. Used to choose the default source and to name the output column.
`worldcover_buffer_m`	Numeric. Buffer radius, in metres, for modal WorldCover extraction. A value of `0` performs point extraction.

soilgrids_sources
: Optional named vector/list of SoilGrids raster paths or URLs. Names are used in output column names.

soilgrids_vars
: Optional character vector of SoilGrids variable names. Used to build default SoilGrids URLs when `soilgrids_sources = NULL`.

soilgrids_buffer_m
: Numeric. Buffer radius, in metres, for mean SoilGrids extraction. A value of `0` performs point extraction.

human_footprint_source
: Optional local path or URL for a Human Footprint raster. When `NULL`, the package default source is used.

human_footprint_buffer_m
: Numeric. Buffer radius, in metres, for mean Human Footprint extraction. A value of `0` performs point extraction.

quiet
: Logical. If `TRUE`, suppress progress messages.

Value

An `sf` object with the same rows and geometry as `sf_points`, with additional raster-context columns appended for the requested contexts. When `"worldcover"` is requested, the returned object includes a `worldcover_<year>` column containing extracted WorldCover class values for the selected `worldcover_year`. When `"soilgrids"` is requested, one column is added for each prepared SoilGrids variable, using names of the form `soilgrids_<variable>`. When `"human_footprint"` is requested, the returned object includes a `human_footprint` column. Raster values are extracted directly at point locations when the relevant buffer distance is `0`; otherwise they are summarised within buffers using modal extraction for WorldCover and mean extraction for SoilGrids and Human Footprint. Empty inputs, non-`sf` inputs or calls with no requested raster contexts are returned unchanged.

Data licensing and attribution

This function can download or extract values from third-party raster products, including WorldCover, SoilGrids and Human Footprint layers. biofetchR does not ship, redistribute or modify these raster datasets. Users are responsible for checking the licence, citation and attribution requirements of each raster source before use in analysis, publication or redistribution. Always cite the specific product, version, year and provider used in the workflow.

See Also

Other raster context helpers: `bf_enrich_raster_context_from_sources()`

Examples

```
if (
  requireNamespace("sf", quietly = TRUE) &&
  requireNamespace("terra", quietly = TRUE)
) {
  gbif_points <- sf::st_as_sf(
    data.frame(
```

```
    species = "Example species",
    decimalLongitude = -5,
    decimalLatitude = 55
  ),
  coords = c("decimalLongitude", "decimalLatitude"),
  crs = 4326,
  remove = FALSE
)

r <- terra::rast(
  nrows = 2,
  ncols = 2,
  xmin = -10,
  xmax = 0,
  ymin = 50,
  ymax = 60,
  crs = "EPSG:4326"
)
terra::values(r) <- c(10, 20, 30, 40)

raster_path <- file.path(tempdir(), "worldcover_example.tif")
terra::writeRaster(r, raster_path, overwrite = TRUE)

enriched <- bf_enrich_raster_context(
  sf_points = gbif_points,
  raster_context = "worldcover",
  cache_dir = tempdir(),
  worldcover_source = raster_path,
  worldcover_year = 2020
)

names(enriched)
}
```

`bf_enrich_raster_context_from_sources` *Enrich point occurrences from explicitly supplied raster context layers*

Description

Adds one or more raster-derived contextual variables to an input point `sf` object using raster sources supplied directly by the user. This helper is intended for biofetchR occurrence-enrichment workflows where environmental or anthropogenic raster covariates need to be attached to GBIF-derived point records before downstream modelling, mapping, filtering or summary steps.

Usage

```
bf_enrich_raster_context_from_sources(
  sf_points,
```

```
  raster_context = c("worldcover", "soilgrids", "human_footprint"),
  cache_dir = NULL,
  worldcover_source = NULL,
  worldcover_year = 2021,
  worldcover_buffer_m = 0,
  soilgrids_sources = NULL,
  soilgrids_vars = NULL,
  soilgrids_buffer_m = 0,
  human_footprint_source = NULL,
  human_footprint_buffer_m = 0,
  force_refresh = FALSE,
  quiet = TRUE
)
```

Arguments

sf_points	Point sf object in any coordinate reference system.
raster_context	Character vector selecting one or more raster contexts. Supported values are "worldcover", "soilgrids" and "human_footprint".
cache_dir	Directory used to cache downloaded raster files when any supplied raster source is a URL. Must be supplied explicitly if URL raster sources are used. In examples, tests and vignettes, use a path under tempdir().
worldcover_source	Path, URL or terra::SpatRaster for the WorldCover raster. Required when "worldcover" is requested.
worldcover_year	Numeric or character label stored in the output worldcover_year column. This is metadata only and does not automatically select or download a raster.
worldcover_buffer_m	Numeric buffer radius in metres for modal WorldCover extraction. Use 0 for direct point extraction.
soilgrids_sources	Named list or named vector of SoilGrids raster sources. Names are used to construct output columns of the form soilgrids_<variable>. Required when "soilgrids" is requested.
soilgrids_vars	Optional character vector of SoilGrids variable names to extract. Defaults to all names in soilgrids_sources.
soilgrids_buffer_m	Numeric buffer radius in metres for mean SoilGrids extraction. Use 0 for direct point extraction.
human_footprint_source	Path, URL or terra::SpatRaster for the Human Footprint raster. Required when "human_footprint" is requested.
human_footprint_buffer_m	Numeric buffer radius in metres for mean Human Footprint extraction. Use 0 for direct point extraction.

force_refresh Logical. If TRUE, re-download URL raster sources even when cached files already exist.

quiet Logical. If TRUE, suppress routine messages and download output where possible.

Details

bf_enrich_raster_context_from_sources() is the source-explicit companion to bf_enrich_raster_context(). Use this function when the raster products are already available locally, when you want to provide custom raster URLs, or when you want to control the exact raster versions used in the analysis.

For package-managed/default raster downloads, use bf_enrich_raster_context() instead.

This function currently supports three optional raster-context groups:

"worldcover" Extracts an ESA WorldCover-style categorical raster, returning worldcover_class, worldcover_label and worldcover_year.

"soilgrids" Extracts one or more named continuous SoilGrids-style rasters and stores them as soilgrids_<variable> columns.

"human_footprint" Extracts a continuous Human Footprint-style raster and stores it as human_footprint.

Raster sources may be local file paths, URLs or pre-loaded terra::SpatRaster objects. URL sources are downloaded into cache_dir and reused on later runs unless force_refresh = TRUE.

Extraction is performed directly at point coordinates when the relevant buffer argument is 0. If a positive buffer radius is supplied, raster values are summarised within buffered geometries: modal class for WorldCover and mean value for SoilGrids and Human Footprint.

Value

An sf object with the same rows and geometry as sf_points, with additional raster-context columns appended for the requested contexts. When "worldcover" is requested, the returned object includes worldcover_class, worldcover_label and worldcover_year. When "soilgrids" is requested, one numeric column is added for each requested SoilGrids variable, using names of the form soilgrids_<variable>. When "human_footprint" is requested, the returned object includes a numeric human_footprint column. Raster values are extracted directly at point locations when the relevant buffer distance is 0; otherwise they are summarised within buffers using modal extraction for categorical WorldCover data and mean extraction for continuous SoilGrids and Human Footprint data. No rows are intentionally added or removed by this enrichment helper.

Output columns

Depending on the requested raster contexts, the returned object may include:

worldcover_class Integer categorical class extracted from the supplied WorldCover-style raster.

worldcover_label Human-readable land-cover label corresponding to worldcover_class, where the class is recognised.

worldcover_year Year or label supplied through worldcover_year.

soilgrids_<variable> Continuous raster values extracted from each named SoilGrids-style raster.

human_footprint Continuous Human Footprint-style raster values.

Data licensing and attribution

This function extracts values from user-supplied raster products, including ESA WorldCover, SoilGrids and Human Footprint-style layers. biofetchR does not ship, redistribute or modify these raster datasets. Users are responsible for checking the licence, citation and attribution requirements of each raster source before use in analysis, publication or redistribution.

Commonly used sources include ESA WorldCover and SoilGrids, which are commonly distributed under Creative Commons Attribution licences, and Human Footprint products, whose terms vary by dataset and provider. Always cite the specific raster product, version, year and provider used in your workflow.

See Also

Other raster context helpers: `bf_enrich_raster_context()`

Examples

```
if (
  requireNamespace("sf", quietly = TRUE) &&
  requireNamespace("terra", quietly = TRUE)
) {
  occ_sf <- sf::st_as_sf(
    data.frame(
      species = "Example species",
      decimalLongitude = -5,
      decimalLatitude = 55
    ),
    coords = c("decimalLongitude", "decimalLatitude"),
    crs = 4326,
    remove = FALSE
  )

  r <- terra::rast(
    nrows = 2,
    ncols = 2,
    xmin = -10,
    xmax = 0,
    ymin = 50,
    ymax = 60,
    crs = "EPSG:4326"
  )
  terra::values(r) <- c(10, 20, 30, 40)

  worldcover_path <- file.path(tempdir(), "worldcover_example.tif")
  footprint_path <- file.path(tempdir(), "human_footprint_example.tif")

  terra::writeRaster(r, worldcover_path, overwrite = TRUE)
  terra::writeRaster(r, footprint_path, overwrite = TRUE)

  enriched <- bf_enrich_raster_context_from_sources(
    sf_points = occ_sf,
    raster_context = c("worldcover", "human_footprint"),
```

```
    cache_dir = tempdir(),
    worldcover_source = worldcover_path,
    human_footprint_source = footprint_path,
    worldcover_year = 2021
  )

  names(enriched)
}
```

`bf_fetch_native_ranges_sinas` *Fetch native-origin evidence from SInAS*

Description

Reads SInAS 3.1.1 native-location records, converts SInAS `locationID` values to ISO3 countries using `AllLocations`, and returns native-origin evidence in the same structure as the optional native web/API helpers. When local paths are not supplied, the function uses `bf_download_sinas_resources()` to resolve package-managed cached copies from Zenodo.

Usage

```
bf_fetch_native_ranges_sinas(
  species,
  species_col = "species",
  cache_dir = NULL,
  force_refresh = FALSE,
  quiet = FALSE,
  main_path = NULL,
  alllocations_path = NULL,
  fulltaxa_path = NULL,
  record_id = "18220953",
  main_url = NULL,
  fulltaxa_url = NULL,
  config_zip_url = NULL,
  return = c("long", "species", "list")
)
```

Arguments

`species`	Character vector of species names, or a data frame containing a species column.
`species_col`	Species column name when `species` is a data frame.
`cache_dir`	Cache directory used when SInAS resources need to be downloaded. Must be supplied explicitly unless all three local resource paths are supplied via `main_path`, `alllocations_path` and `fulltaxa_path`. In examples, tests and vignettes, use a path under `tempdir()`.

force_refresh	Logical; re-download/re-read package-managed resources.
quiet	Logical; suppress progress messages.
main_path	Optional local path to SInAS_3.1.1.csv.
alllocations_path	Optional local path to AllLocations.xlsx, .csv, or .tsv.
fulltaxa_path	Optional local path to SInAS_3.1.1_FullTaxaList.csv.
record_id	Zenodo record identifier used by bf_download_sinas_resources().
main_url	Optional direct URL for the main SInAS CSV.
fulltaxa_url	Optional direct URL for the FullTaxaList CSV.
config_zip_url	Optional direct URL for the config archive containing AllLocations.xlsx.
return	One of "long", "species" or "list".

Details

This is a **provider-specific** helper. It extracts native-origin evidence from SInAS and returns species-level origin countries, but it does not decide whether a recipient country is native or non-native. Pass the returned species-level table to bf_attach_native_status() for recipient-level classification.

SInAS 3.1.1 can use a quoted whitespace-delimited text format despite the .csv extension. The internal reader therefore tries a SInAS-aware parser before falling back to more conventional CSV, TSV and pipe-delimited readers.

Value

The returned object depends on return. If return = "long", the function returns a tibble with one row per retained SInAS native-origin evidence record, including species, source, accepted_name, source_taxon_id, raw_native_area, raw_status, origin_iso3, evidence_type and source_url. If return = "species", the function returns a species-level tibble with collapsed native-origin evidence, including species, native_origin_iso3, native_sources_used, native_web_unmapped_strings, native_web_n_records and native_has_origin. If return = "list", the function returns a named list with long, species, unmapped and summary elements. The unmapped element contains long-format evidence rows without resolved ISO3 origin codes, and summary is a one-row tibble reporting input species counts, matched SInAS records, ISO3-resolved species and unmapped records. These outputs provide species-level native-origin evidence only; recipient- level native/non-native classification is performed by bf_attach_native_status().

Relationship to utils_native_range.R

Keep this script alongside utils_native_range.R. This file provides the SInAS evidence source; utils_native_range.R provides the general standardisation, attachment, filtering and reconciliation logic used by the terrestrial/freshwater and marine pipelines.

Data source and attribution

biofetchR resolves and standardises SInAS resources but does not own or redistribute the underlying data. Users should cite the exact SInAS release, Zenodo record, version and access date used in their workflow, and should follow the licence terms attached to that release.

See Also

bf_attach_native_status(), bf_download_sinas_resources()

Examples

```
if (interactive()) {
  sinas_native <- bf_fetch_native_ranges_sinas(
    species = c("Carcinus maenas", "Ficopomatus enigmaticus"),
    cache_dir = file.path(tempdir(), "biofetchR_sinas"),
    return = "species",
    quiet = FALSE
  )

  sinas_native
}
```

---

bf_fetch_native_ranges_web

Compile species-level native-origin evidence from web sources

---

Description

Queries one or more web/API sources for species-level native-origin evidence, caches raw responses, converts source locality/country strings to ISO3 where possible, and returns both long evidence and species-level collapsed outputs.

Usage

```
bf_fetch_native_ranges_web(
  species,
  species_col = "species",
  sources = c("sinas", "gbif", "worms"),
  cache_dir = NULL,
  force_refresh = FALSE,
  sleep_sec = 0.25,
  quiet = FALSE,
  sinas_main_path = NULL,
  sinas_alllocations_path = NULL,
  sinas_fulltaxa_path = NULL,
  sinas_record_id = "18220953",
  sinas_main_url = NULL,
  sinas_fulltaxa_url = NULL,
  sinas_config_zip_url = NULL,
  return = c("list", "species", "long")
)
```

Arguments

species	Character vector of species names, or a data frame containing a species column.
species_col	Species column name when species is a data frame.
sources	Character vector of native evidence sources. Currently supports "sinas", "gbif" and "worms".
cache_dir	Directory used for all web-response caches. Must be supplied explicitly when web/API sources need to be queried or cached. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; if TRUE, ignore cached responses and fetch from web sources again.
sleep_sec	Delay between requests, in seconds.
quiet	Logical; suppress progress messages.
sinas_main_path	Optional local path to SInAS_3.1.1.csv.
sinas_alllocations_path	Optional local path to AllLocations.xlsx, .csv or .tsv.
sinas_fulltaxa_path	Optional local path to SInAS_3.1.1_FullTaxaList.csv.
sinas_record_id	Zenodo record identifier used for package-managed SInAS downloads.
sinas_main_url	Optional direct URL for the main SInAS CSV.
sinas_fulltaxa_url	Optional direct URL for the FullTaxaList CSV.
sinas_config_zip_url	Optional direct URL for the SInAS config archive.
return	One of "list", "species" or "long".

Details

This helper is designed as an optional complement to curated native-range tables. Its collapsed species output can be passed to bf_attach_native_status() as native_ranges, because it includes a species column and a semicolon-delimited native_origin_iso3 column.

Value

The returned object depends on return. If return = "list", the function returns a named list with four elements: long, a tibble with one row per retained source evidence record using the standard columns species, source, accepted_name, source_taxon_id, raw_native_area, raw_status, origin_iso3, evidence_type and source_url; species, a collapsed species-level tibble containing species, native_origin_iso3, native_sources_used, native_web_unmapped_strings, native_web_n_records and native_has_origin; unmapped, the subset of long-format native-like evidence rows whose locality or area string could not be resolved to ISO3; and summary, a one-row tibble reporting input species counts, matched web records, ISO3-resolved species, unmapped records and sources used. If return = "species", only the collapsed species-level tibble is returned. If return = "long", only the long evidence tibble is returned. These outputs provide species-level native-origin evidence for downstream use in bf_attach_native_status() and do not themselves classify recipient records as native or non-native.

Relationship to SInAS

SInAS support is now part of this canonical web-source aggregator. When `sources` includes `"sinas"`, this function delegates provider-specific parsing to `bf_fetch_native_ranges_sinas()`. Keep that SInAS provider file in the package, but do not also keep a duplicate `*_with_sinas` aggregator file.

Data source and interpretation

The returned evidence is web/API-derived and source-dependent. Treat it as an auditable native-origin evidence layer rather than proof that a species is native, non-native or absent from any place.

See Also

Other native-range web-source helpers: `bf_available_native_web_sources()`, `bf_web_native_gbif()`, `bf_web_native_worms()`, `bf_write_native_web_outputs()`

Examples

```
if (interactive()) {
  native_web <- bf_fetch_native_ranges_web(
    species = c("Carcinus maenas", "Ficopomatus enigmaticus"),
    sources = c("gbif", "worms"),
    cache_dir = file.path(tempdir(), "biofetchR_native_web"),
    return = "species",
    quiet = FALSE
  )

  native_web
}
```

---

bf_filter_griis_invasive

Keep rows flagged as invasive in GRIIS

Description

Convenience wrapper around `bf_attach_griis_status()` that attaches GRIIS evidence and then retains only rows where `griis_invasive` is `TRUE`.]

Usage

```
bf_filter_griis_invasive(
  df,
  species_col = "species",
  iso2c_col = NULL,
  iso3c_col = NULL,
```

```
  country_col = NULL,
  griis = NULL,
  cache_dir = NULL,
  force_refresh = FALSE,
  require_country = TRUE,
  quiet = FALSE
)
```

Arguments

df	Input data frame.
species_col	Name of the column containing species names.
iso2c_col	Optional name of an ISO2 recipient-country column.
iso3c_col	Optional name of an ISO3 recipient-country column.
country_col	Optional name of a recipient country-name column.
griis	Optional raw or standardised GRIIS table. If NULL, GRIIS is read with bf_read_griis().
cache_dir	Cache directory used when griis = NULL. Must be supplied explicitly when the helper needs to download/read GRIIS internally. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; if TRUE, re-download/re-read GRIIS when the helper loads it.
require_country	Logical. If TRUE, filter by species + recipient-country GRIIS evidence. If FALSE, filter by species-level GRIIS evidence.
quiet	Logical; if TRUE, suppress progress messages.

Value

A tibble containing the subset of input rows for which attached GRIIS evidence has griis_invasive == TRUE. The returned object includes the original input columns plus the GRIIS evidence columns added by bf_attach_griis_status(), including griis_listed, griis_invasive, griis_status and supporting matched-name/status metadata where available. Rows excluded by this filter are not necessarily native or impact-free; they simply lack a matching invasive GRIIS flag under the selected matching mode.

Interpretation

This is a conservative convenience filter based only on GRIIS invasive-status evidence available under the selected matching mode. Rows removed by the filter are not necessarily native or impact-free; they simply lack a matching invasive GRIIS flag under the supplied criteria.

See Also

Other GRIIS origin-evidence helpers: bf_attach_griis_status(), bf_download_griis(), bf_find_griis_table(), bf_griis_lookup(), bf_read_griis(), bf_standardise_griis(), bf_unpack_griis()

Examples

```
raw <- data.frame(
  species = "Example species",
  scientificName = "Example species",
  countryCode_alpha2 = "GB",
  countryCode_alpha3 = "GBR",
  isInvasive = "yes",
  stringsAsFactors = FALSE
)
input <- data.frame(species = "Example species", iso2c = "GB")
bf_filter_griis_invasive(input, iso2c_col = "iso2c", griis = raw)
```

`bf_filter_native` *Filter a data frame to confirmed native recipient records*

Description

Filter a data frame to confirmed native recipient records

Usage

```
bf_filter_native(
  df,
  species_col = "species",
  iso2c_col = NULL,
  iso3c_col = NULL,
  country_col = NULL,
  native_ranges,
  native_species_col = "species",
  origin_iso3_col = NULL,
  require_country = TRUE,
  strategy = c("tiered", "union"),
  max_origins = 100L,
  quiet = FALSE
)
```

Arguments

`df`	Input data frame.
`species_col`	Species column in `df`.
`iso2c_col`	Optional ISO2 recipient column in `df`.
`iso3c_col`	Optional ISO3 recipient column in `df`.
`country_col`	Optional recipient country-name column in `df`. Used only when `require_country = TRUE` and ISO2/ISO3 columns are unavailable.

native_ranges Native-origin evidence table.

native_species_col
: Species column in native_ranges.

origin_iso3_col
: Optional origin ISO3 column in native_ranges.

require_country
: Logical; when TRUE, classify recipient status using recipient ISO2/ISO3. When FALSE, attach species-level origin availability.

strategy Native-origin column strategy passed to bf_native_range_lookup().

max_origins Maximum origin count per species.

quiet Logical; suppress messages.

Value

A tibble containing only rows classified by bf_attach_native_status() as confirmed native recipients under the supplied native-origin evidence and recipient-country settings. The returned object includes the original input columns plus the native-origin/status columns added by bf_attach_native_status(), including native_is_native_recipient, native_status, native_filter_keep and native_filter_rejection_reasc Rows excluded by this filter are not necessarily non-native; they simply lack confirmed native recipient evidence under the selected criteria.

See Also

Other native-origin status helpers: bf_attach_native_status(), bf_filter_non_native(), bf_native_range_lookup(bf_native_status_summary(), bf_reconcile_griis_native_status(), bf_standardise_native_ranges()

---

bf_filter_non_native *Filter a data frame to confirmed non-native recipient records*

Description

Filter a data frame to confirmed non-native recipient records

Usage

```
bf_filter_non_native(
  df,
  species_col = "species",
  iso2c_col = NULL,
  iso3c_col = NULL,
  country_col = NULL,
  native_ranges,
  native_species_col = "species",
  origin_iso3_col = NULL,
  require_country = TRUE,
  strategy = c("tiered", "union"),
```

```
  max_origins = 100L,
  quiet = FALSE
)
```

Arguments

df	Input data frame.
species_col	Species column in df.
iso2c_col	Optional ISO2 recipient column in df.
iso3c_col	Optional ISO3 recipient column in df.
country_col	Optional recipient country-name column in df. Used only when require_country = TRUE and ISO2/ISO3 columns are unavailable.
native_ranges	Native-origin evidence table.
native_species_col	Species column in native_ranges.
origin_iso3_col	Optional origin ISO3 column in native_ranges.
require_country	Logical; when TRUE, classify recipient status using recipient ISO2/ISO3. When FALSE, attach species-level origin availability.
strategy	Native-origin column strategy passed to bf_native_range_lookup().
max_origins	Maximum origin count per species.
quiet	Logical; suppress messages.

Value

A tibble containing only rows classified by bf_attach_native_status() as confirmed non-native recipients under the supplied native-origin evidence and recipient-country settings. The returned object includes the original input columns plus the native-origin/status columns added by bf_attach_native_status(), including native_is_non_native_recipient, native_status, native_filter_keep and native_filter_rejection_r Rows excluded by this filter are not necessarily native; they simply lack confirmed non-native recipient evidence under the selected criteria.

See Also

Other native-origin status helpers: bf_attach_native_status(), bf_filter_native(), bf_native_range_lookup(), bf_native_status_summary(), bf_reconcile_griis_native_status(), bf_standardise_native_ranges()

`bf_find_griis_table` *Find the most likely GRIIS table in an unpacked archive*

Description

Searches an unpacked GRIIS directory for CSV, TSV, TXT or TAB files and selects the most likely data table using filename and file-size heuristics. Metadata, readme and citation files are down-weighted.

Usage

```
bf_find_griis_table(exdir)
```

Arguments

`exdir`	Directory returned by `bf_unpack_griis()`.

Value

A character vector of length one giving the path to the selected GRIIS table file within `exdir`. The selected file is chosen from CSV, TSV, TXT or TAB files using filename and file-size heuristics that favour likely species or compendium data tables and down-weight metadata, readme and citation files.

See Also

Other GRIIS origin-evidence helpers: `bf_attach_griis_status()`, `bf_download_griis()`, `bf_filter_griis_invasive()`, `bf_griis_lookup()`, `bf_read_griis()`, `bf_standardise_griis()`, `bf_unpack_griis()`

Examples

```
exdir <- tempfile("griis_example_")
dir.create(exdir)

write.csv(
  data.frame(species = "Example species"),
  file.path(exdir, "griis_country_compendium.csv"),
  row.names = FALSE
)

table_path <- bf_find_griis_table(exdir)
basename(table_path)
```

bf_griis_lookup *Build a compact GRIIS lookup table*

Description

Creates a deduplicated lookup table from a raw or standardised GRIIS table. The lookup can be built either at species x country resolution or at species-only resolution.

Usage

```
bf_griis_lookup(griis, require_country = TRUE)
```

Arguments

griis	Raw or standardised GRIIS table.
require_country	Logical. If TRUE, build a species x ISO3 lookup. If FALSE, build a species-only lookup.

Details

Use require_country = TRUE for terrestrial and freshwater species-country workflows, where invasive status should be matched against the recipient country. Use require_country = FALSE for marine or global workflows that only need species-level GRIIS evidence before GBIF download submission.

The lookup combines accepted-name and scientific-name keys, then summarises all matching GRIIS rows into a compact evidence table.

Value

A deduplicated tibble used as a compact GRIIS lookup table. When require_country = TRUE, each row represents a species-name key by ISO3 country-code combination. When require_country = FALSE, each row represents species-level GRIIS evidence without country matching. The output includes griis_listed, griis_invasive, griis_any_invasive_field_present, griis_n_matches and collapsed supporting fields such as matched names, countries, establishment means, occurrence status and taxon rank. These fields summarise matching GRIIS evidence for downstream joins and filters.

Interpretation

The lookup reports whether a supplied name/country combination matches GRIIS evidence and whether any matched GRIIS row is flagged invasive. Non-matches should be treated as not_listed_or_no_match, not as confirmed absence of introduction or impact.

See Also

Other GRIIS origin-evidence helpers: bf_attach_griis_status(), bf_download_griis(), bf_filter_griis_invasive
bf_find_griis_table(), bf_read_griis(), bf_standardise_griis(), bf_unpack_griis()

Examples

```
raw <- data.frame(
  species = "Example species",
  scientificName = "Example species",
  countryCode_alpha2 = "GB",
  countryCode_alpha3 = "GBR",
  isInvasive = "yes",
  stringsAsFactors = FALSE
)
lookup <- bf_griis_lookup(raw, require_country = TRUE)
lookup
```

---

bf_load_basins *Load nested basins (HydroBASINS; Pfafstetter)*

Description

Downloads selected regional tiles at a chosen Pfafstetter level (L1-L12), caches once, and returns polygons with canonical fields: hybas_id, next_down, next_sink, main_bas, sub_area, up_area, endo, order.

Usage

```
bf_load_basins(level = 12, with_lakes = FALSE, cache_dir = NULL, quiet = TRUE)
```

Arguments

level	Integer from 1 to 12 giving the Pfafstetter basin level.
with_lakes	Logical; if TRUE, use the customised HydroBASINS "with lakes" variant.
cache_dir	Cache root used for downloaded HydroBASINS files. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
quiet	Logical; if TRUE, suppress download and status messages.

Value

An sf polygon object containing HydroBASINS catchments for the requested Pfafstetter level, in WGS84 longitude/latitude coordinates. The returned object includes canonical basin attributes such as hybas_id, next_down, next_sink, main_bas, sub_area, up_area, endo and order, where available. Each row represents a basin polygon used for assigning occurrence records to freshwater catchment units.

See Also

Other freshwater overlay loaders: bf_load_feow(), bf_load_lakes(), bf_load_rivers(), bf_name_rivers_osm()

`bf_load_biosphere_reserve` *Load UNESCO biosphere reserve locations*

Description

Downloads, caches and reads a UNESCO biosphere reserve CSV, then converts recognised longitude/latitude columns to an `sf` point object. The function is remote-first but accepts local or alternate remote sources for reproducible workflows and tests.

Usage

```
bf_load_biosphere_reserve(
  cache_dir = NULL,
  force_refresh = FALSE,
  iso2c = NULL,
  clip_to_countries = TRUE,
  source_path = NULL,
  source_url = NULL,
  quiet = TRUE
)
```

Arguments

`cache_dir`	Character. Directory used for the downloaded CSV cache. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`force_refresh`	Logical. If `TRUE`, re-download the CSV even when a cache file is already present.
`iso2c`	Optional character vector of ISO2 country codes used to clip the returned layer when `clip_to_countries = TRUE`.
`clip_to_countries`	Logical. If `TRUE` and `iso2c` is supplied, keep only points intersecting the requested countries.
`source_path`	Optional local CSV path. Used instead of downloading when the path exists.
`source_url`	Optional remote CSV URL. Used instead of the package default source.
`quiet`	Logical. If `TRUE`, suppress progress messages.

Value

An `sf` point object in WGS84 longitude/latitude coordinates containing UNESCO biosphere reserve locations. The returned object includes `biosphere_id`, a stable reserve or row identifier; `biosphere_name`, a human-readable reserve label where available; the original longitude and latitude columns used to create point geometry; and the active geometry column. If `clip_to_countries = TRUE` and `iso2c` is supplied, the returned object is filtered to biosphere reserve points intersecting the requested countries where possible.

Data licensing and attribution

biofetchR does not redistribute UNESCO biosphere reserve data. Users are responsible for checking and following the licence, citation and attribution requirements of the source used in their workflow.

See Also

Other remote overlay loaders: `bf_load_gdw_barriers()`, `bf_load_gloric()`

Examples

```
if (interactive()) {
  biosphere <- bf_load_biosphere_reserve(
    iso2c = "FR",
    cache_dir = file.path(tempdir(), "biofetchR_biosphere"),
    clip_to_countries = TRUE
  )

  biosphere
}
```

---

bf_load_feow *Load Freshwater Ecoregions of the World polygons*

Description

Loads the Freshwater Ecoregions of the World (FEOW) polygon layer, standardises key fields, and caches the result as a single GeoPackage. The default `method = "auto"` first attempts a direct ArcGIS FeatureServer query, then falls back to a downloadable archive if available.

Usage

```
bf_load_feow(
  force_refresh = FALSE,
  quiet = TRUE,
  cache_dir = NULL,
  method = "auto",
  source_url = NULL,
  ...
)
```

Arguments

`force_refresh`	Logical. If `TRUE`, ignore any existing cached FEOW GeoPackage and rebuild the cache.
`quiet`	Logical. If `TRUE`, suppress progress messages.

cache_dir	Cache root used for the FEOW cache. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
method	Character. One of "auto", "arcgis", "download" or "download". "auto" tries the methods in that order.
source_url	Optional source URL used when method = "download". If omitted, the FEOW downloads page is queried for a ZIP URL.
...	Ignored. Retained for backward compatibility with older calls.

Value

An sf polygon object containing Freshwater Ecoregions of the World features in WGS84 longitude/latitude coordinates. The returned object includes standardised columns such as feow_id, ecoregion, biome and realm, where these fields are available from the source layer. Each row represents a freshwater ecoregion polygon used for assigning occurrence records to freshwater spatial units.

Data access and attribution

FEOW is a third-party freshwater ecoregion product. biofetchR only downloads and caches the layer locally for the user. Users should cite FEOW and check the provider's licence and redistribution requirements before using outputs in publications or shared data products.

See Also

Other freshwater overlay loaders: bf_load_basins(), bf_load_lakes(), bf_load_rivers(), bf_name_rivers_osm()

Examples

```
if (interactive()) {
  feow <- bf_load_feow(
    cache_dir = file.path(tempdir(), "biofetchR_feow"),
    quiet = TRUE
  )

  names(feow)
}
```

---

bf_load_gdw_barriers *Load Global Dam Watch river barrier points*

Description

Downloads, caches and reads the Global Dam Watch barrier layer, then returns a standardised point sf object. By default, the function uses the public Figshare article metadata to locate the Global Dam Watch v1.0 shapefile archive. Local files or alternate URLs can be supplied with source_path or source_url for testing, mirrors or manually downloaded data.

Usage

```
bf_load_gdw_barriers(
  cache_dir = NULL,
  force_refresh = FALSE,
  iso2c = NULL,
  clip_to_countries = TRUE,
  source_path = NULL,
  source_url = NULL,
  quiet = TRUE
)
```

Arguments

cache_dir	Character. Directory used for downloaded archives, extracted files and metadata caches. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical. If TRUE, re-download remote files and rebuild extracted caches where possible.
iso2c	Optional character vector of ISO2 country codes used to clip the returned layer when clip_to_countries = TRUE.
clip_to_countries	Logical. If TRUE and iso2c is supplied, keep only features intersecting the requested countries.
source_path	Optional local vector file or archive path. Used instead of downloading when the path exists.
source_url	Optional remote URL to a vector file or archive. Used instead of the package default source.
quiet	Logical. If TRUE, suppress progress messages.

Value

An sf point object in WGS84 longitude/latitude coordinates containing Global Dam Watch barrier features. The returned object includes gdw_barrier_id, a stable barrier, dam or feature identifier; gdw_barrier_name, a human-readable barrier or dam label where available; and the active geometry column. If the selected source layer is not already point geometry, centroids are returned so occurrence points can be assigned to nearby barrier features by downstream workflows. If clip_to_countries = TRUE and iso2c is supplied, the returned object is filtered to the requested countries where possible. The function stops if no rows remain after processing or clipping.

Data licensing and attribution

biofetchR does not redistribute Global Dam Watch data. Users are responsible for checking and following the licence, citation and attribution requirements of the specific Global Dam Watch release used in their workflow.

See Also

Other remote overlay loaders: bf_load_biosphere_reserve(), bf_load_gloric()

Examples

```
if (interactive()) {
  gdw <- bf_load_gdw_barriers(
    iso2c = c("GB", "IE"),
    cache_dir = file.path(tempdir(), "biofetchR_gdw_barriers"),
    clip_to_countries = TRUE
  )

  gdw
}
```

`bf_load_gdw_reservoirs` *Load Global Dam Watch reservoir polygons*

Description

Downloads, caches, reads, and standardises Global Dam Watch reservoir polygon data for use as a terrestrial/freshwater context overlay. The returned object contains stable columns used by the GBIF processing pipeline: `gdw_reservoir_id`, `gdw_reservoir_name`, and geometry.

Usage

```
bf_load_gdw_reservoirs(
  cache_dir = NULL,
  force_refresh = FALSE,
  iso2c = NULL,
  clip_to_countries = TRUE,
  source_path = NULL,
  source_url = NULL,
  quiet = TRUE
)
```

Arguments

`cache_dir`	Directory used for downloads, extracted files, and cached source data. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`force_refresh`	Logical; re-download and re-read cached files.
`iso2c`	Optional ISO2 country code vector used to crop the layer.
`clip_to_countries`	Logical; if `TRUE` and `iso2c` is supplied, crop the returned features to those countries.
`source_path`	Optional local source-file or directory override.
`source_url`	Optional remote source URL override.
`quiet`	Logical; suppress messages where supported.

Value

An sf polygon object in WGS84 longitude/latitude coordinates containing Global Dam Watch reservoir features. The returned object includes gdw_reservoir_id, a stable reservoir identifier; gdw_reservoir_name, a human-readable reservoir or dam label where available; and the active geometry column. Each row represents one reservoir polygon or multipolygon used for assigning occurrence records to reservoir context. If clip_to_countries = TRUE and iso2c is supplied, the returned object is cropped to the requested countries where possible.

Data access and attribution

This loader can download Global Dam Watch reservoir data from provider-hosted metadata/download services or read a user-supplied local/source URL. Users should cite the Global Dam Watch data release and associated publication for the exact version used. biofetchR caches the source locally for reproducible processing but does not redistribute Global Dam Watch data.

See Also

Other additional context overlay loaders: bf_load_global_mining(), bf_load_hydrowaste()

Examples

```
if (interactive()) {
  reservoirs <- bf_load_gdw_reservoirs(
    iso2c = "GB",
    cache_dir = file.path(tempdir(), "biofetchR_gdw_reservoirs"),
    clip_to_countries = TRUE
  )

  reservoirs
}
```

---

bf_load_global_mining *Load global mining polygons*

Description

Downloads, caches, reads, and standardises global mining polygon data for use as a terrestrial context overlay. The returned object contains stable columns used by the GBIF processing pipeline: global_mining_id, global_mining_name, and geometry.

Usage

```
bf_load_global_mining(
  cache_dir = NULL,
  force_refresh = FALSE,
  iso2c = NULL,
```

```
  clip_to_countries = TRUE,
  source_path = NULL,
  source_url = NULL,
  quiet = TRUE
)
```

Arguments

cache_dir	Directory used for downloads, extracted files, and cached source data. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
force_refresh	Logical; re-download and re-read cached files.
iso2c	Optional ISO2 country code vector used to crop the layer.
clip_to_countries	Logical; if `TRUE` and `iso2c` is supplied, crop the returned features to those countries.
source_path	Optional local source-file or directory override.
source_url	Optional remote source URL override.
quiet	Logical; suppress messages where supported.

Value

An `sf` polygon object in WGS84 longitude/latitude coordinates containing global mining-area features. The returned object includes `global_mining_id`, a stable mining-feature identifier; `global_mining_name`, a human-readable mine, country or feature label where available; and the active geometry column. Each row represents one mining polygon or multipolygon used for assigning occurrence records to mining-area context. If `clip_to_countries = TRUE` and `iso2c` is supplied, the returned object is cropped to the requested countries where possible.

Data access and attribution

This loader can download the global-scale mining polygons from PANGAEA or read a user-supplied local/source URL. Users should cite the PANGAEA dataset DOI, version and associated publication and follow the dataset licence attached to the downloaded file. biofetchR caches the source locally for reproducible processing but does not redistribute the mining dataset.

See Also

Other additional context overlay loaders: `bf_load_gdw_reservoirs()`, `bf_load_hydrowaste()`

Examples

```
if (interactive()) {
  mines <- bf_load_global_mining(
    iso2c = "ZA",
    cache_dir = file.path(tempdir(), "biofetchR_global_mining"),
    clip_to_countries = TRUE
  )
```

```
  mines
}
```

`bf_load_gloric` *Load GloRiC river reach lines*

Description

Downloads, caches and reads the Global River Classification (GloRiC) line layer, then standardises core identifier/class columns for use as a terrestrial/freshwater context overlay.

Usage

```
bf_load_gloric(
  cache_dir = NULL,
  force_refresh = FALSE,
  iso2c = NULL,
  clip_to_countries = TRUE,
  source_path = NULL,
  source_url = NULL,
  quiet = TRUE
)
```

Arguments

`cache_dir`	Character. Directory used for downloaded archives and extracted files. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`force_refresh`	Logical. If `TRUE`, re-download remote files and rebuild extracted caches where possible.
`iso2c`	Optional character vector of ISO2 country codes used to clip the returned layer when `clip_to_countries = TRUE`.
`clip_to_countries`	Logical. If `TRUE` and `iso2c` is supplied, crop the river layer to the requested country boundaries.
`source_path`	Optional local vector file or archive path. Used instead of downloading when the path exists.
`source_url`	Optional remote URL to a vector file or archive. Used instead of the package default source.
`quiet`	Logical. If `TRUE`, suppress progress messages.

Value

An `sf` line object in WGS84 longitude/latitude coordinates containing GloRiC river-reach features. The returned object includes `gloric_id`, a stable reach identifier; `gloric_class`, the selected GloRiC reach-type or classification label where available; and the active geometry column. Each row represents one river reach line or multiline feature used as terrestrial/freshwater context. If `clip_to_countries = TRUE` and `iso2c` is supplied, the returned object is cropped to the requested countries where possible. The function stops if no rows remain after processing or clipping.

Data licensing and attribution

biofetchR does not redistribute GloRiC data. Users are responsible for checking and following the licence, citation and attribution requirements of the GloRiC version used in their workflow.

See Also

Other remote overlay loaders: `bf_load_biosphere_reserve()`, `bf_load_gdw_barriers()`

Examples

```
if (interactive()) {
  rivers <- bf_load_gloric(
    iso2c = "GB",
    cache_dir = file.path(tempdir(), "biofetchR_gloric"),
    clip_to_countries = TRUE
  )

  rivers
}
```

---

bf_load_hydrowaste *Load HydroWASTE wastewater treatment plant points*

Description

Downloads, caches, reads, and standardises the HydroWASTE wastewater treatment plant layer for use as a terrestrial/freshwater context overlay. The returned object contains stable columns used by the GBIF processing pipeline: `hydrowaste_id`, `hydrowaste_name`, `hydrowaste_level`, and geometry.

Usage

```
bf_load_hydrowaste(
  cache_dir = NULL,
  force_refresh = FALSE,
  iso2c = NULL,
  clip_to_countries = TRUE,
```

```
  source_path = NULL,
  source_url = NULL,
  quiet = TRUE
)
```

Arguments

cache_dir	Directory used for downloads, extracted files, and cached source data. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; re-download and re-read cached files.
iso2c	Optional ISO2 country code vector used to crop the layer.
clip_to_countries	Logical; if TRUE and iso2c is supplied, crop the returned features to those countries.
source_path	Optional local source-file or directory override.
source_url	Optional remote source URL override.
quiet	Logical; suppress messages where supported.

Details

HydroWASTE is treated as a point overlay. If the source is not already point geometry, centroids are used so occurrence points can be assigned to the nearest treatment-plant feature by the main pipeline.

Value

An sf point object in WGS84 longitude/latitude coordinates containing HydroWASTE wastewater-treatment plant features. The returned object includes hydrowaste_id, a stable treatment-plant identifier; hydrowaste_name, a human-readable plant or feature label; hydrowaste_level, the treatment-level attribute where available; and the active geometry column. If the source layer is not already point geometry, centroids are returned so occurrence points can be assigned to the nearest HydroWASTE feature by downstream workflows. If clip_to_countries = TRUE and iso2c is supplied, the returned object is cropped to the requested countries where possible.

Data access and attribution

This loader can download HydroWASTE v1.0 from a package-managed public mirror or read a user-supplied local/source URL. HydroWASTE should be cited using the original dataset publication and any provider guidance associated with the exact file/version used. biofetchR caches the source locally for reproducible processing but does not redistribute HydroWASTE.

See Also

Other additional context overlay loaders: bf_load_gdw_reservoirs(), bf_load_global_mining()

Examples

```
if (interactive()) {
  hydrowaste <- bf_load_hydrowaste(
    iso2c = c("GB", "IE"),
    cache_dir = file.path(tempdir(), "biofetchR_hydrowaste"),
    clip_to_countries = TRUE
  )

  hydrowaste
}
```

`bf_load_lakes` *Load global lake polygons (HydroLAKES)*

Description

Downloads HydroLAKES once (RAW), standardizes key fields, then applies optional filters in-memory on each call. Cached as a single GeoPackage.

Usage

```
bf_load_lakes(
  force_refresh = FALSE,
  quiet = TRUE,
  cache_dir = NULL,
  lakes_only = TRUE,
  min_area_km2 = NULL
)
```

Arguments

`force_refresh`	logical; re-download even if cache exists.
`quiet`	logical; suppress messages.
`cache_dir`	Cache root used for the HydroLAKES cache. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`lakes_only`	logical; keep only natural lakes (exclude reservoirs/controls).
`min_area_km2`	optional numeric; drop lakes smaller than this area.

Value

An `sf` polygon object containing HydroLAKES features in WGS84 longitude/latitude coordinates. The returned object includes standardised columns `lake_id`, `name`, `is_reservoir` and `area_km2`, alongside any retained source attributes. Each row represents a lake polygon. If `lakes_only` or `min_area_km2` is used, the returned object contains only the lakes retained by those filters.

See Also

Other freshwater overlay loaders: bf_load_basins(), bf_load_feow(), bf_load_rivers(), bf_name_rivers_osm()

---

bf_load_marine_regions_overlay

Load a package-managed Marine Regions overlay

Description

Loads one supported marine spatial overlay, caches the source product locally, and returns a standardised sf object for use in the marine GBIF processing pipeline.

Usage

```
bf_load_marine_regions_overlay(
  region_source = "eez",
  cache_dir = NULL,
  force_refresh = FALSE,
  quiet = TRUE
)
```

Arguments

region_source	Character. Marine overlay name or alias. See bf_available_marine_overlays(include_extended = TRUE) for registered options.
cache_dir	Directory used for Marine Regions product caches. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical. If TRUE, ignore the local RDS cache and re-download/reload the product through mregions2.
quiet	Logical. If TRUE, suppress progress messages.

Details

The loader first resolves region_source to a canonical biofetchR overlay name, then attempts to load the corresponding Marine Regions product through mregions2. Candidate product identifiers are tried in order because product names may differ between Marine Regions releases.

The returned object is transformed to EPSG:4326, repaired where possible and standardised to include:

- marine_region_id
- marine_region_name
- marine_region_source
- marine_regions_layer
- geoname

These columns are used by process_gbif_marine_pipeline() when assigning occurrence records to marine spatial units and exporting grouped CSV outputs.

Value

An sf polygon object in WGS84 longitude/latitude coordinates containing the requested Marine Regions overlay. The returned object includes canonical columns used by downstream marine workflows: marine_region_id, a character identifier for each spatial unit; marine_region_name, a character region label; marine_region_source, the canonical biofetchR overlay name; marine_regions_layer, the Marine Regions product identifier used to load the data; and geoname, a standardised human-readable region name. Source attributes may also be retained after these canonical columns. Each row represents one marine polygon or multipolygon used for assigning occurrence records to marine contextual regions. The source overlay is cached locally as an RDS file for reuse.

Caching

Loaded overlays are cached as RDS files under cache_dir using one subfolder per canonical overlay. Set force_refresh = TRUE to ignore the cached object and request a fresh copy through mregions2.

Data source, licensing and attribution

Marine overlays are retrieved from Marine Regions data products through mregions2. biofetchR only standardises and caches those products for local processing; it does not redistribute them as package data. Users should cite Marine Regions and the specific product loaded, using the marine_regions_layer output column and the license / citation fields in mregions2::mrp_list where available.

See Also

Other marine overlay loaders: bf_available_marine_overlays(), bf_marine_overlay_canonical()

Examples

```
if (interactive()) {
  meow <- bf_load_marine_regions_overlay(
    region_source = "meow",
    cache_dir = file.path(tempdir(), "biofetchR_marine_regions")
  )

  meow
}
```

---

bf_load_ne_admin1 *Load Natural Earth Admin-1 (States/Provinces) (auto-download + cache)*

Description

Creates:

- id_ne_admin1: stable polygon identifier
- geoname_ne_admin1: admin-1 name

Usage

```
bf_load_ne_admin1(
  cache_dir,
  scale = c("50m", "10m"),
  force_refresh = FALSE,
  clip_to_countries = TRUE,
  iso2c = NULL,
  quiet = TRUE,
  url = NULL
)
```

Arguments

cache_dir	Character. Folder to store Natural Earth downloads. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
scale	Character. "50m" (smaller) or "10m" (more detailed).
force_refresh	Logical. If TRUE, re-download and re-extract.
clip_to_countries	Logical. If TRUE, crop to bbox of iso2c (if provided).
iso2c	Optional ISO2 code vector for cropping.
quiet	Logical. If TRUE, reduce messages.
url	Optional override URL.

Value

An sf polygon object containing Natural Earth Admin-1 state/province features in WGS84 longitude/latitude coordinates. The returned object contains id_ne_admin1, a character identifier for each administrative unit; geoname_ne_admin1, a character state/province name where available from the source layer or a generated fallback label otherwise; and the active geometry column. Each row represents one first-level administrative polygon or multipolygon used for assigning occurrence records to subnational Natural Earth administrative units. If clip_to_countries = TRUE and iso2c is supplied, the object is filtered to the requested countries where country-identifying attributes are available, with a bounding-box crop used as a fallback.

Data source and attribution

Downloads Natural Earth Admin-1 state/province polygons from the Natural Earth public distribution endpoint unless url is overridden. biofetchR caches and standardises the layer but does not redistribute or own the data. Cite Natural Earth and the scale used in any downstream analysis.

See Also

Other Natural Earth and RESOLVE overlay loaders: `bf_load_ne_urban()`, `bf_load_resolve2017()`

`bf_load_ne_urban` *Load Natural Earth Urban Areas (auto-download + cache)*

Description

Creates two columns:

- `id_ne_urban`: stable ID per polygon
- `geoname_ne_urban`: urban area name (if available)

Usage

```
bf_load_ne_urban(
  cache_dir,
  scale = c("10m"),
  force_refresh = FALSE,
  clip_to_countries = TRUE,
  iso2c = NULL,
  quiet = TRUE,
  url = NULL
)
```

Arguments

`cache_dir`	Character. Folder to store downloaded zips and extracted shapefiles. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`scale`	Character. Natural Earth scale, typically "10m".
`force_refresh`	Logical. If TRUE, re-download and re-extract.
`clip_to_countries`	Logical. If TRUE, crop to bbox of `iso2c` (if provided).
`iso2c`	Optional character vector of ISO2 country codes used for cropping.
`quiet`	Logical. If TRUE, reduce messages.
`url`	Optional override URL.

Value

An `sf` polygon object containing Natural Earth urban-area features in WGS84 longitude/latitude coordinates. The returned object contains `id_ne_urban`, a character identifier assigned by biofetchR; `geoname_ne_urban`, a character urban-area name where available from the source layer or a generated fallback label otherwise; and the active geometry column. Each row represents one urban-area polygon or multipolygon used for assigning occurrence records to Natural Earth urban-context overlays. If `clip_to_countries = TRUE` and `iso2c` is supplied, the object is cropped to a buffered bounding box around the requested countries.

Data source and attribution

Downloads Natural Earth urban-area polygons from the Natural Earth public distribution endpoint unless `url` is overridden. biofetchR caches and standardises the layer but does not redistribute or own the data. Cite Natural Earth and the scale used in any downstream analysis.

See Also

Other Natural Earth and RESOLVE overlay loaders: `bf_load_ne_admin1()`, `bf_load_resolve2017()`

`bf_load_ne_urban_areas` *Load Natural Earth Urban Areas (polygons), auto-downloaded and cached*

Description

This downloads the Natural Earth "Urban Areas" polygons (10m cultural vectors), caches the zip, extracts it, and returns an sf object in EPSG:4326 with a consistent ID + label column for downstream point-tagging.

Usage

```
bf_load_ne_urban_areas(
  cache_dir = NULL,
  url = "http://www.naturalearthdata.com/download/10m/cultural/ne_10m_urban_areas.zip",
  force_refresh = FALSE,
  quiet = TRUE
)
```

Arguments

`cache_dir`	Character. Cache directory root. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`url`	Character. Optional override download URL.
`force_refresh`	Logical. Re-download and re-extract even if cached.
`quiet`	Logical. Reduce messages.

Details

Users do not need to pre-download shapefiles; biofetchR handles retrieval and caching.

Value

An `sf` polygon object containing Natural Earth urban-area features in WGS84 longitude/latitude coordinates. The returned object contains `urban_id`, a character identifier for each feature; `urban_name`, a character label for the urban area; and the active geometry column. Each row represents one urban-area polygon or multipolygon used for assigning occurrence records to urban contextual overlays.

See Also

Other terrestrial overlay loaders: bf_load_resolve_ecoregions2017(), bf_load_teow(), bf_teow_cache_info(), bf_teow_clear_cache()

`bf_load_ramsar` *Load Ramsar wetland polygons for selected countries*

Description

Downloads, standardises and caches Ramsar-designated polygon features for one or more countries. Ramsar features are obtained as a designation subset of WDPA, which keeps provenance, licensing, geometry handling and update cadence consistent with the WDPA overlay workflow.

Usage

```
bf_load_ramsar(
  iso2c,
  cache_dir = NULL,
  force_refresh = FALSE,
  quiet = TRUE,
  require_opt_in = TRUE,
  opt_in = FALSE
)
```

Arguments

iso2c	Character vector of ISO 3166-1 alpha-2 country codes.
cache_dir	Character. Directory used to read/write cached country extracts. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical. If TRUE, ignore existing cache files and re-download country extracts.
quiet	Logical. If FALSE, print cache/download progress messages.
require_opt_in	Logical. If TRUE, require opt_in = TRUE before any download is attempted.
opt_in	Logical. Must be TRUE when require_opt_in = TRUE.

Details

The function is deliberately opt-in because Ramsar features are queried through the WDPA/Protected Planet service. By default, require_opt_in = TRUE and users must explicitly set opt_in = TRUE after confirming that their intended use complies with the relevant terms. Cached files are written as one GeoPackage per ISO3 country code.

Value

An sf polygon object in WGS84 longitude/latitude coordinates containing Ramsar wetland features for the requested countries. The returned object uses the standard biofetchR Ramsar schema: ramsar_id, a stable Ramsar or WDPA-derived site identifier; ramsar_name, the Ramsar site name; ramsar_iso3, the ISO3 country code associated with the feature; ramsar_desig_eng, the English designation label where available; and the active geometry column. If no valid country codes are supplied, or if no Ramsar features are returned for the requested countries, a zero-row sf object with the same standard columns is returned.

Data licensing

Ramsar records returned here are derived from the WDPA/Protected Planet service and remain subject to the relevant external licensing and attribution requirements. Do not redistribute downloaded geometries through biofetchR, package examples, tests, or derived cache folders.

See Also

Other overlay loaders: bf_load_wdpa()

Examples

```
if (interactive()) {
  ramsar_gb <- bf_load_ramsar(
    iso2c = "GB",
    cache_dir = file.path(tempdir(), "biofetchR_ramsar"),
    opt_in = TRUE,
    quiet = FALSE
  )

  ramsar_gb
}
```

---

bf_load_resolve2017 *Load RESOLVE Ecoregions (2017) (auto-download + cache)*

Description

RESOLVE ecoregions provide a modern, consistent terrestrial biogeographic layer that is frequently used in biodiversity and macroecology workflows. We access the UNEP-WCMC ArcGIS service, download a country-bbox subset (by default), cache it as a GeoPackage, and return it as an sf object ready for point-in-polygon tagging.

Usage

```
bf_load_resolve2017(
  cache_dir,
  force_refresh = FALSE,
  clip_to_countries = TRUE,
  iso2c = NULL,
  quiet = TRUE,
  service_url = NULL,
  max_allowable_offset = 0.01,
  geometry_precision = 5,
  chunk_size = 200
)
```

Arguments

cache_dir	Character. Cache directory. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical. Re-download even if cached.
clip_to_countries	Logical. If TRUE and iso2c provided, query only bbox around those countries.
iso2c	Optional ISO2 codes for bbox query when clip_to_countries=TRUE.
quiet	Logical. Reduce messages.
service_url	Optional character URL. If NULL, the default RESOLVE provider URL is used.
max_allowable_offset	Numeric. ArcGIS geometry simplification in degrees (outSR=4326).
geometry_precision	Integer. ArcGIS geometry precision (decimal places).
chunk_size	Integer. Pagination chunk size.

Details

Output columns:

- id_resolve2017 (stable ID)
- geoname_resolve2017 (ecoregion name)
- biome_resolve2017 (biome name if available)
- realm_resolve2017 (realm if available)

Value

An sf polygon object containing RESOLVE 2017 terrestrial ecoregion features in WGS84 longitude/latitude coordinates, or NULL when the provider query returns no readable features for the requested extent. When features are returned, the object contains id_resolve2017, a character ecoregion identifier; geoname_resolve2017, a character ecoregion name; biome_resolve2017, the biome name where available; realm_resolve2017, the realm name where available; and the active geometry column. Each row represents one ecoregion polygon or multipolygon used for assigning occurrence records to RESOLVE terrestrial ecoregion context. The returned subset is also cached locally as a GeoPackage for reuse.

Data source and attribution

Queries the UNEP-WCMC ArcGIS FeatureServer for the RESOLVE 2017 terrestrial ecoregions layer and caches the returned subset as a GeoPackage. biofetchR does not redistribute or own the service data. Cite the RESOLVE ecoregions data source/service and record the access date and query settings used.

See Also

Other Natural Earth and RESOLVE overlay loaders: `bf_load_ne_admin1()`, `bf_load_ne_urban()`

`bf_load_resolve_ecoregions2017` *Load RESOLVE Ecoregions 2017 (polygons), auto-downloaded and cached*

Description

Downloads the RESOLVE terrestrial ecoregions dataset (2017 update), caches and extracts it, and returns an sf object in EPSG:4326 with consistent ID + label columns.

Usage

```
bf_load_resolve_ecoregions2017(
  cache_dir = NULL,
  url = "https://storage.googleapis.com/teow2016/Ecoregions2017.zip",
  force_refresh = FALSE,
  quiet = TRUE
)
```

Arguments

`cache_dir`	Character. Cache directory root. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`url`	Character. Optional override download URL.
`force_refresh`	Logical. Re-download and re-extract even if cached.
`quiet`	Logical. Reduce messages.

Details

Users do not need to pre-download shapefiles; biofetchR handles retrieval and caching.

Value

An `sf` polygon object containing RESOLVE 2017 terrestrial ecoregion features in WGS84 longitude/latitude coordinates. The returned object contains `resolve_eco_id`, a character ecoregion identifier; `resolve_eco_name`, a character ecoregion label; `resolve_biome`, the biome name where available; `resolve_realm`, the realm name where available; and the active geometry column. Each row represents one ecoregion polygon or multipolygon used for terrestrial contextual assignment.

See Also

Other terrestrial overlay loaders: bf_load_ne_urban_areas(), bf_load_teow(), bf_teow_cache_info(), bf_teow_clear_cache()

Examples

```
if (interactive()) {
  resolve <- bf_load_resolve_ecoregions2017(
    cache_dir = file.path(tempdir(), "biofetchR_resolve_ecoregions")
  )

  names(resolve)
}
```

---

bf_load_rivers *Load global river reaches (HydroRIVERS) with optional caching*

Description

Downloads + standardizes HydroRIVERS. If cache="disk", writes an RDS cache (and optionally a GPKG). If cache="memory", returns in-memory only. Filters (min_strahler, min_discharge_cms) are applied on read, not baked into cache.

Usage

```
bf_load_rivers(
  force_refresh = FALSE,
  quiet = TRUE,
  cache_dir = NULL,
  cache = c("disk", "memory"),
  cache_format = c("rds", "gpkg", "both"),
  regions = NULL,
  min_strahler = NULL,
  min_discharge_cms = NULL
)
```

Arguments

force_refresh	logical; re-download even if cache exists (disk mode).
quiet	logical; suppress messages.
cache_dir	Cache root used when cache = "disk". Must be supplied explicitly for disk caching. Ignored when cache = "memory". In examples, tests and vignettes, use a path under tempdir().
cache	"disk" or "memory". Disk persists a cache; memory does not write files.

cache_format	"rds","gpkg","both" (used only when cache="disk"; default "rds").
regions	Character vector. Supported values include "global", "af", "ar", "as", "au", "eu", "gr", "na", "sa", and "si". If NULL, "global" is used.
min_strahler	optional integer; keep ORD_STRA >= this (applied on read).
min_discharge_cms	optional numeric; keep DIS_AV_CMS >= this (applied on read).

Value

An sf line object containing HydroRIVERS reaches in WGS84 longitude/latitude coordinates. The returned object includes canonical reach attributes hyriv_id, ord_stra, ord_clas, ord_flow, dis_av_cms, length_km and hybas_l12. Each row represents a river reach. If min_strahler or min_discharge_cms is supplied, the returned object contains only reaches retained by those filters.

See Also

Other freshwater overlay loaders: bf_load_basins(), bf_load_feow(), bf_load_lakes(), bf_name_rivers_osm()

---

bf_load_teow *Load & cache WWF Terrestrial Ecoregions (TEOW)*

Description

Downloads (once) and caches the WWF terrestrial ecoregions as an sf object. Prefer method = "mapme" if the **mapme.biodiversity** package is installed; otherwise, use method = "direct" with a source_url to the official TEOW ZIP.

Usage

```
bf_load_teow(
  cache_dir = NULL,
  method = c("auto", "mapme", "direct"),
  source_url = NULL,
  aoi = NULL,
  bbox = NULL,
  force_refresh = FALSE,
  simplify = FALSE,
  simplify_tolerance = 0.01,
  quiet = FALSE
)
```

Arguments

cache_dir	Cache root used for the TEOW cache. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
method	Character. One of "auto", "mapme" or "direct".
source_url	Optional direct URL to a TEOW ZIP file when method = "direct".
aoi	Optional sf or sfc object used to clip the TEOW layer.
bbox	Optional bounding box used to clip the TEOW layer.
force_refresh	Logical; if TRUE, rebuild the TEOW cache even if it already exists.
simplify	Logical; if TRUE, simplify TEOW geometries after loading.
simplify_tolerance	Numeric simplification tolerance used when simplify = TRUE.
quiet	Logical; if TRUE, suppress progress messages.

Details

The cached, normalized layer is written to <cache_dir>/teow/teow.gpkg with the ecoregion name exposed as geoname and the geometry column standardized to "geometry".

Value

An sf polygon object containing Terrestrial Ecoregions of the World features in WGS84 longitude/latitude coordinates. The returned object includes a standard geoname column and, where available, original ecoregion, biome and realm attributes. If aoi or bbox is supplied, the returned layer is clipped to that area; otherwise it contains the cached or freshly loaded global layer. The normalised layer is also cached locally as a GeoPackage for reuse.

Data access and attribution

TEOW is a third-party terrestrial ecoregion product. biofetchR downloads and caches it locally for user-side analysis only. Users should cite the original provider and check licence and redistribution terms before sharing derived products.

See Also

Other terrestrial overlay loaders: bf_load_ne_urban_areas(), bf_load_resolve_ecoregions2017(), bf_teow_cache_info(), bf_teow_clear_cache()

Examples

```
if (interactive()) {
  teow <- bf_load_teow(
    cache_dir = file.path(tempdir(), "biofetchR_teow"),
    quiet = TRUE
  )

  names(teow)
}
```

bf_load_wdpa *Load WDPA protected-area polygons for selected countries*

Description

Downloads, standardises and caches country-level WDPA protected-area polygons from the UNEP-WCMC/Protected Planet licensed ArcGIS service. biofetchR does not redistribute WDPA data; this function creates local user-side cache files that can be reused in later overlay runs.

Usage

```
bf_load_wdpa(
  iso2c,
  cache_dir = NULL,
  force_refresh = FALSE,
  quiet = TRUE,
  exclude_marine = TRUE,
  extra_where = NULL,
  require_opt_in = TRUE,
  opt_in = FALSE
)
```

Arguments

iso2c	Character vector of ISO 3166-1 alpha-2 country codes, for example "GB" or c("GB", "IE").
cache_dir	Character. Directory used to read/write cached country extracts. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical. If TRUE, ignore existing cache files and re-download country extracts.
quiet	Logical. If FALSE, print cache/download progress messages.
exclude_marine	Logical. If TRUE, exclude WDPA features where realm == "Marine" in the ArcGIS query.
extra_where	Optional character string containing an additional ArcGIS SQL filter to append to the query with AND. Intended for advanced use.
require_opt_in	Logical. If TRUE, require opt_in = TRUE before any download is attempted.
opt_in	Logical. Must be TRUE when require_opt_in = TRUE.

Details

The function is deliberately opt-in because WDPA data are licensed. By default, require_opt_in = TRUE and users must explicitly set opt_in = TRUE after confirming that their intended use complies with WDPA/Protected Planet terms. Cached files are written as one GeoPackage per ISO3 country code.

When exclude_marine = TRUE, features whose WDPA realm is exactly "Marine" are excluded from the service query. Mixed or terrestrial records are retained.

Value

An `sf` polygon object in WGS84 longitude/latitude coordinates containing WDPA protected-area features for the requested countries. The returned object uses the standard biofetchR WDPA schema: `wdpa_id`, a stable protected-area identifier; `wdpa_name`, the protected-area name; `wdpa_desig_eng`, the English designation label where available; `wdpa_iucn_cat`, the IUCN protected-area category where available; `wdpa_iso3`, the ISO3 country code associated with the feature; `wdpa_area_km2`, the reported GIS area in square kilometres where available; and the active geometry column. If no valid country codes are supplied, or if no features are returned for the requested countries, a zero-row `sf` object with the same standard columns is returned.

Data licensing

WDPA/Protected Planet data are subject to external licensing and attribution requirements. Do not redistribute downloaded WDPA geometries through biofetchR, package examples, tests, or derived cache folders.

See Also

Other overlay loaders: `bf_load_ramsar()`

Examples

```
if (interactive()) {
  wdpa_gb <- bf_load_wdpa(
    iso2c = "GB",
    cache_dir = file.path(tempdir(), "biofetchR_wdpa"),
    opt_in = TRUE,
    quiet = FALSE
  )

  wdpa_gb
}
```

`bf_marine_overlay_canonical` *Resolve a marine overlay alias to its canonical name*

Description

Converts a user-supplied overlay name or accepted alias to the canonical biofetchR overlay name used internally.

Usage

```
bf_marine_overlay_canonical(region_source)
```

Arguments

region_source	Character. Marine overlay name or alias, for example "eez", "iho_seas" or "longhurst_provinces".

Details

Marine Regions product names can vary across data releases and package versions. biofetchR therefore keeps a registry of accepted aliases. This function performs the alias lookup and returns the canonical name used by the marine pipeline and overlay loader.

Value

A character vector of length one containing the canonical biofetchR marine overlay name corresponding to region_source. The returned value is one of the registered overlay identifiers used internally by the marine pipeline and by bf_load_marine_regions_overlay(). The function errors if region_source cannot be matched to a supported overlay or alias.

See Also

Other marine overlay loaders: bf_available_marine_overlays(), bf_load_marine_regions_overlay()

Examples

```
bf_marine_overlay_canonical("eez")
bf_marine_overlay_canonical("iho_seas")
```

---

bf_name_rivers_osm *Annotate HydroRIVERS reaches with OSM river names*

Description

Annotate HydroRIVERS reaches with OSM river names

Usage

```
bf_name_rivers_osm(rivers, iso2c, max_snap_m = 500, quiet = TRUE)
```

Arguments

rivers	sf LINESTRING with field hyriv_id (from bf_load_rivers()).
iso2c	character ISO2 country codes used to build a bbox query.
max_snap_m	numeric, max distance (m) to accept a name match.
quiet	logical.

Value

An sf line object with the same river reaches as rivers, returned in WGS84 longitude/latitude coordinates, with an added character column river_name. The river_name column contains the nearest accepted OpenStreetMap waterway name within max_snap_m, or NA_character_ when no suitable named waterway is found. The output is used to add human-readable river labels to HydroRIVERS features.

See Also

Other freshwater overlay loaders: bf_load_basins(), bf_load_feow(), bf_load_lakes(), bf_load_rivers()

---

bf_native_range_lookup

Build a native-range lookup table

Description

Thin wrapper around bf_standardise_native_ranges() retained for readability in pipelines and tests.

Usage

```
bf_native_range_lookup(
  native_ranges,
  species_col = "species",
  origin_iso3_col = NULL,
  strategy = c("tiered", "union"),
  max_origins = 100L,
  quiet = FALSE
)
```

Arguments

native_ranges	Data frame containing native-origin evidence to standardise.
species_col	Species column in native_raw.
origin_iso3_col	Optional explicit ISO3-origin column.
strategy	Origin-column strategy. Currently "tiered" and "union" both collapse all available ISO3 origin columns conservatively.
max_origins	Maximum number of ISO3 origins to retain per species.
quiet	Logical; suppress messages.

Value

A tibble with the same structure as bf_standardise_native_ranges(): one row per standardised species-name key and columns describing species-level native-origin evidence, including native_name, native_name_key, native_origin_iso3, native_origin_flag, native_has_origin and native_sources_used. This function is a wrapper around bf_standardise_native_ranges() and returns the lookup table used by downstream native-status joins.

See Also

Other native-origin status helpers: bf_attach_native_status(), bf_filter_native(), bf_filter_non_native(), bf_native_status_summary(), bf_reconcile_griis_native_status(), bf_standardise_native_ranges()

---

bf_native_status_summary

Summarise native-origin status columns

Description

Summarise native-origin status columns

Usage

```
bf_native_status_summary(x)
```

Arguments

x	Data frame returned by bf_attach_native_status().

Value

A tibble summarising the native_status column in x. The output contains one row per native-status class and two columns: native_status, the character status label, and n, the number of rows assigned to that status. Rows are ordered from the most frequent status to the least frequent, then alphabetically by status label. The summary is intended for quick auditing of native-origin classification outcomes.

See Also

Other native-origin status helpers: bf_attach_native_status(), bf_filter_native(), bf_filter_non_native(), bf_native_range_lookup(), bf_reconcile_griis_native_status(), bf_standardise_native_ranges()

bf_prepare_taxa_for_gbif *Prepare taxon names for GBIF download pipelines*

Description

Applies manual fixes first, then cleans names, removes obvious non-taxa, blocks genus-level/open/abbreviated names, and returns accepted, rejected, audit and summary tables.

Usage

```
bf_prepare_taxa_for_gbif(
  df,
  name_col = "species",
  iso2_col = NULL,
  manual_fixes = NULL,
  require_species_level = TRUE,
  deduplicate = TRUE
)
```

Arguments

df	Input data frame.
name_col	Column containing raw taxon names.
iso2_col	Optional ISO2 country column. Use "iso2c" for terrestrial/freshwater workflows.
manual_fixes	Optional named character vector or data frame passed to bf_apply_manual_taxonomy_fixes().
require_species_level	Logical. If TRUE, only species-level candidates are accepted.
deduplicate	Logical. If TRUE, accepted names are deduplicated.

Value

A named list with four elements: accepted, a tibble/data frame of cleaned taxon records retained for GBIF download workflows; rejected, a tibble/data frame of records removed before GBIF submission with rejection diagnostics; audit, a tibble recording raw names, fixed names, cleaned names, genus values, acceptance flags and rejection reasons for every input row; and summary, a one-row tibble with counts and percentages describing the preparation outcome. The output separates records suitable for download from records rejected as non-taxonomic, open-name or non-species-level candidates.

bf_read_griis *Read and standardise the GRIIS country compendium*

Description

Downloads or reuses the cached GRIIS country compendium, reads the selected table and converts it to biofetchR's standard GRIIS schema. This is the main entry point used by the pipeline origin-evidence gates when a pre-read GRIIS table is not supplied.

Usage

```
bf_read_griis(cache_dir = NULL, force_refresh = FALSE, quiet = FALSE)
```

Arguments

cache_dir	Directory used for the cached GRIIS file.
force_refresh	Logical; if TRUE, re-download and re-read the GRIIS source.
quiet	Logical; if TRUE, suppress progress messages.

Value

A tibble returned by bf_standardise_griis() containing the standardised GRIIS country-compendium fields used by biofetchR. The output includes stable species-name keys, country identifiers, ISO2 and ISO3 codes, invasive-status evidence and optional taxonomic/status metadata. Each row represents a standardised GRIIS species-country evidence record.

Data source and attribution

This helper reads the external GRIIS country compendium after download/caching and standardises it for biofetchR joins. Users remain responsible for citing the GRIIS source, version/record and any associated data publication in downstream outputs.

See Also

Other GRIIS origin-evidence helpers: bf_attach_griis_status(), bf_download_griis(), bf_filter_griis_invasive bf_find_griis_table(), bf_griis_lookup(), bf_standardise_griis(), bf_unpack_griis()

Examples

```
if (interactive()) {
  griis <- bf_read_griis(
    cache_dir = file.path(tempdir(), "biofetchR_griis"),
    quiet = FALSE
  )

  names(griis)
}
```

bf_reconcile_griis_native_status

Reconcile GRIIS and native-origin evidence

Description

Reconcile GRIIS and native-origin evidence

Usage

```
bf_reconcile_griis_native_status(x)
```

Arguments

x	Data frame containing GRIIS columns and native-status columns.

Value

A tibble containing the input rows with one additional character column, `biofetchr_origin_evidence_status`. The added column summarises agreement, conflict or missingness between available GRIIS evidence and native-origin recipient evidence, with labels such as `"griis_invasive_native_origin_supported"`, `"griis_invasive_native_origin_conflict"`, `"native_origin_non_native_only"` and `"origin_unknown"`. The returned table preserves the input columns and should be interpreted as an audit table for evidence consistency, not as an independent biological classification.

Interpretation

Reconciliation labels describe agreement or conflict between available GRIIS and native-origin evidence. They are audit labels, not independent biological proof of invasion status.

See Also

Other native-origin status helpers: `bf_attach_native_status()`, `bf_filter_native()`, `bf_filter_non_native()`, `bf_native_range_lookup()`, `bf_native_status_summary()`, `bf_standardise_native_ranges()`

bf_resolve_gbif_taxonomy

Resolve one taxon name against GBIF

Description

Resolve one taxon name against GBIF

Usage

```
bf_resolve_gbif_taxonomy(
  name,
  strict = FALSE,
  use_gbif_parse = TRUE,
  use_gbif_suggest = TRUE,
  use_genus_fallback = TRUE,
  use_worms_fallback = FALSE,
  manual_overrides = NULL,
  retries = 3,
  sleep_sec = 0.15,
  quiet = TRUE
)
```

Arguments

name	Taxon name.
strict	Passed to rgbif::name_backbone().
use_gbif_parse	Use rgbif::name_parse() for canonicalisation.
use_gbif_suggest	Try rgbif::name_suggest() if backbone fails.
use_genus_fallback	If TRUE, retry a genus-level match for open names.
use_worms_fallback	If TRUE and {worrms} is installed, use WoRMS as an external fallback for unresolved names/ranks. This does not provide a GBIF taxonKey but can fill rank fields.
manual_overrides	Optional named vector or data frame of manual name fixes.
retries	Number of retries for remote calls.
sleep_sec	Delay between failed retries.
quiet	Reduce messages.

Value

A one-row tibble describing the taxonomy-resolution outcome for name. The output contains the original and cleaned names, the query name, GBIF usage keys, the taxon key selected for downloads, resolved scientific and canonical names, rank, status, match type, confidence, standard classification columns from kingdom to species, resolver provenance, taxon_match_status, any warning text, and logical flags for genus-level or non-taxonomic inputs. Unresolved or skipped names are returned in the same schema with missing key fields and diagnostic status values.

`bf_resolve_gbif_taxonomy_batch` *Resolve a vector of taxon names against GBIF*

Description

Resolve a vector of taxon names against GBIF

Usage

```
bf_resolve_gbif_taxonomy_batch(
  names,
  cache_dir = NULL,
  cache_file = "bf_gbif_taxonomy_cache.rds",
  refresh_cache = FALSE,
  strict = FALSE,
  strict_taxonomy = FALSE,
  quiet = TRUE,
  ...
)
```

Arguments

`names`	Character vector of taxon names.
`cache_dir`	Optional cache directory. If supplied, an RDS cache is read and written there. If `NULL`, no taxonomy cache is read or written. In examples, tests and vignettes, use a path under `tempdir()` when caching is needed.
`cache_file`	Cache file name.
`refresh_cache`	Ignore existing cached rows.
`strict`	Passed to `rgbif::name_backbone()`.
`strict_taxonomy`	If TRUE, stop if any non-empty/non-junk name is unresolved.
`quiet`	Reduce messages.
`...`	Passed to `bf_resolve_gbif_taxonomy()`.

Value

A tibble with one row per input name, preserving input order and using the same taxonomy-resolution columns returned by `bf_resolve_gbif_taxonomy()`. The output includes cleaned names, GBIF keys, resolved names, rank and classification fields, resolver provenance, diagnostic status columns and flags for genus-level or non-taxonomic inputs. When a cache is used, cached and newly resolved rows are combined into the same output schema.

`bf_sinas_default_urls` *Return default SInAS 3.1.1 resource URLs*

Description

Constructs the direct Zenodo file URLs used by `bf_download_sinas_resources()`. The default record is the SInAS 3.1.1 record used by the biofetchR native-evidence workflow.

Usage

```
bf_sinas_default_urls(record_id = "18220953")
```

Arguments

`record_id`	Character scalar. Zenodo record identifier used to construct the file URLs. Defaults to `"18220953"`.

Value

A named character vector of direct Zenodo file URLs used by the SInAS downloader. The vector contains four entries: `main_csv`, the URL for `SInAS_3.1.1.csv`; `config_zip`, the URL for `All_Config_Files_SInAS_v3.1.1.zip`, which contains `AllLocations.xlsx`; `output_zip`, the URL for `All_Output_Files_SInAS_v3.1.1.zip`, which contains `SInAS_3.1.1_FullTaxaList.csv` for the default record; and `fulltaxa_csv`, a legacy-style standalone FullTaxaList URL returned for completeness. The function only constructs URLs and does not download, cache, read or validate any SInAS files.

Data access

This function only constructs URLs. It does not download, cache, read or redistribute SInAS data.

See Also

Other SInAS resource helpers: `bf_download_sinas_resources()`

Examples

```
bf_sinas_default_urls()
```

`bf_standardise_griis` *Standardise a raw GRIIS table*

Description

Converts a raw country-level GRIIS table into the stable column names used by biofetchR. The function identifies accepted/scientific name fields, country/ISO fields, invasive-status fields and optional taxonomic metadata using flexible column-name matching.

Usage

```
bf_standardise_griis(griis_raw)
```

Arguments

`griis_raw`	Raw data frame read from a GRIIS source.

Details

The returned table includes both accepted-name and scientific-name join keys. Downstream lookup construction uses both keys so that records can match either the accepted GRIIS name or a reported scientific-name field.

This helper deliberately checks that the table resembles the country-level GRIIS compendium. If required country-level fields are missing, it stops rather than silently producing a species-only table with empty country or invasive-status evidence.

Value

A tibble with stable `griis_*` columns used by biofetchR lookup and origin-evidence workflows. The output includes `griis_scientific_name`, `griis_accepted_name`, `griis_name_key`, `griis_scientific_name_key`, `griis_country`, `griis_iso2c`, `griis_iso3c` and `griis_is_invasive`, plus optional occurrence, establishment and taxonomic metadata where available. Each row represents a standardised GRIIS species-country evidence record. The invasive-status fields should be interpreted as GRIIS evidence, not as proof of absence where no match exists.

Interpretation

Standardised columns are evidence fields derived from GRIIS records. They should be interpreted as GRIIS-listed status evidence, not as proof that a species is absent from countries where no match is found.

See Also

Other GRIIS origin-evidence helpers: `bf_attach_griis_status()`, `bf_download_griis()`, `bf_filter_griis_invasive`
`bf_find_griis_table()`, `bf_griis_lookup()`, `bf_read_griis()`, `bf_unpack_griis()`

Examples

```
raw <- data.frame(
  species = "Example species",
  scientificName = "Example species",
  countryCode_alpha2 = "GB",
  countryCode_alpha3 = "GBR",
  isInvasive = "yes",
  stringsAsFactors = FALSE
)
std <- bf_standardise_griis(raw)
names(std)
```

`bf_standardise_native_ranges` *Standardise native-range evidence to a species-level lookup*

Description

Converts one or more native-origin ISO3 columns into a compact, one-row-per- species lookup table. This is the normal entry point for provider outputs from SInAS/GBIF/WoRMS helpers or for user-supplied native-range tables.

Usage

```
bf_standardise_native_ranges(
  native_raw,
  species_col = "species",
  origin_iso3_col = NULL,
  strategy = c("tiered", "union"),
  max_origins = 100L,
  quiet = FALSE
)
```

Arguments

`native_raw`	Data frame containing native-origin evidence.
`species_col`	Species column in `native_raw`.
`origin_iso3_col`	Optional explicit ISO3-origin column.
`strategy`	Origin-column strategy. Currently `"tiered"` and `"union"` both collapse all available ISO3 origin columns conservatively.
`max_origins`	Maximum number of ISO3 origins to retain per species.
`quiet`	Logical; suppress messages.

Details

The output is species-level evidence only. Recipient-level native/non-native classification is performed by bf_attach_native_status(), which compares the recipient ISO3 code against the native-origin ISO3 set.

Value

A tibble with one row per standardised species-name key. The output contains native_name, the cleaned species label; native_name_key, the normalised name used for joins; native_origin_iso3, a semicolon-delimited character string of ISO3 native-origin country codes; native_origin_flag, a character flag indicating whether usable origin evidence was found; native_has_origin, a logical indicator of origin-evidence availability; and native_sources_used, a semicolon-delimited provenance field where source information is available. The table represents species-level native-origin evidence only; recipient-level native/non-native classification is performed by bf_attach_native_status().

See Also

Other native-origin status helpers: bf_attach_native_status(), bf_filter_native(), bf_filter_non_native(), bf_native_range_lookup(), bf_native_status_summary(), bf_reconcile_griis_native_status()

---

bf_taxonomic_summary *Summarise taxonomy resolution outcomes*

Description

Summarise taxonomy resolution outcomes

Usage

```
bf_taxonomic_summary(taxonomy_tbl)
```

Arguments

taxonomy_tbl	Output from bf_resolve_gbif_taxonomy_batch().

Value

A one-row tibble summarising taxonomy-resolution outcomes. The output contains n_input, n_matched, n_unresolved, n_non_taxon, n_higher_rank, pct_matched, resolved_by and match_types. These fields report the number of input rows, how many received usable GBIF taxon keys, how many were unresolved or skipped, and which resolver and match-type categories were present.

bf_tax_extract_genus *Extract the genus component from a taxon name*

Description

Extract the genus component from a taxon name

Usage

```
bf_tax_extract_genus(x)
```

Arguments

x	Character vector.

Value

A character vector with the same length as x. Each element contains the extracted genus name from the corresponding cleaned taxon string, or NA_character_ when no plausible genus component can be identified. The output is used for genus-level diagnostics and fallback taxonomy matching.

bf_tax_is_genus_level *Detect genus-level or open-nomenclature names*

Description

Detect genus-level or open-nomenclature names

Usage

```
bf_tax_is_genus_level(x)
```

Arguments

x	Character vector.

Value

A logical vector with the same length as x. Values are TRUE for names that appear to be genus-level, higher-level or open-nomenclature strings, including single-token genus names, sp./spp. names, cf./aff. names, groups, complexes or abbreviated epithets. Values are FALSE for names that are not flagged by these rules.

Flag genus-level, higher-level, or open-name taxonomic strings

bf_tax_looks_non_taxon *Detect likely non-taxonomic strings*

Description

Flags common environmental variables, habitat descriptors, measurement names, and other strings that should not be submitted to taxonomy APIs.

Usage

```
bf_tax_looks_non_taxon(x)
```

Arguments

x	Character vector.

Value

A logical vector with the same length as x. Values are TRUE for missing values, placeholders, measurement-like strings, environmental variables or other inputs that are unlikely to be biological taxon names. Values are FALSE for strings that remain plausible taxonomic names after the conservative screening rules.

Detect strings that are unlikely to be biological taxon names

bf_teow_cache_info *Return cache path/info for TEOW*

Description

Return cache path/info for TEOW

Usage

```
bf_teow_cache_info(cache_dir = NULL)
```

Arguments

cache_dir	Cache root containing the TEOW cache. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().

Value

A named list describing the Terrestrial Ecoregions of the World cache. The list contains `path`, a character string giving the expected GeoPackage cache path; `exists`, a logical value indicating whether the cache file is present; `size_bytes`, a numeric file size in bytes or `NA_real_` when the file is absent; and `modified`, a `POSIXct` modification time or `NA` when the file is absent. The values are intended for cache diagnostics and do not load the spatial layer itself.

See Also

Other terrestrial overlay loaders: `bf_load_ne_urban_areas()`, `bf_load_resolve_ecoregions2017()`, `bf_load_teow()`, `bf_teow_clear_cache()`

`bf_teow_clear_cache` *Clear the cached TEOW dataset*

Description

Clear the cached TEOW dataset

Usage

```
bf_teow_clear_cache(cache_dir = NULL, ask = interactive())
```

Arguments

`cache_dir`	Cache root containing the TEOW cache to remove. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`ask`	If TRUE, prompt before deleting. Set FALSE for non-interactive use.

Value

Invisibly returns `TRUE` when the Terrestrial Ecoregions of the World cache directory does not exist or is successfully removed. Invisibly returns `FALSE` when deletion is cancelled after an interactive prompt. The function is called primarily for its side effect of removing cached TEOW files.

See Also

Other terrestrial overlay loaders: `bf_load_ne_urban_areas()`, `bf_load_resolve_ecoregions2017()`, `bf_load_teow()`, `bf_teow_cache_info()`

bf_unpack_griis *Unpack a cached GRIIS archive*

Description

Unpacks a downloaded GRIIS archive into a cache directory. This helper is kept for archive-style GRIIS sources, although the default biofetchR source is now a country-level CSV that can be read directly without unpacking.

Usage

```
bf_unpack_griis(
  archive_path,
  exdir = NULL,
  force_refresh = FALSE,
  quiet = FALSE
)
```

Arguments

archive_path	Path returned by bf_download_griis() or another downloaded GRIIS archive file.
exdir	Directory to unpack into. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; if TRUE, remove and recreate exdir before unpacking.
quiet	Logical; if TRUE, suppress progress messages.

Value

A character vector of length one giving the normalised path to the directory containing the unpacked GRIIS archive, using forward slashes. If the archive has already been unpacked and force_refresh = FALSE, the existing directory path is returned. The function is also called for the side effect of creating or refreshing the unpacked cache directory.

See Also

Other GRIIS origin-evidence helpers: bf_attach_griis_status(), bf_download_griis(), bf_filter_griis_invasive bf_find_griis_table(), bf_griis_lookup(), bf_read_griis(), bf_standardise_griis()

Examples

```
archive_root <- file.path(tempdir(), "biofetchR_griis_unpack_example")
unlink(archive_root, recursive = TRUE, force = TRUE)
dir.create(archive_root, recursive = TRUE, showWarnings = FALSE)

archive_src <- file.path(archive_root, "src")
dir.create(archive_src, recursive = TRUE, showWarnings = FALSE)
```

```
griis_file <- file.path(archive_src, "griis_country_compendium.csv")

write.csv(
  data.frame(species = "Example species"),
  griis_file,
  row.names = FALSE
)

archive_path <- file.path(archive_root, "griis_example.tar")

utils::tar(
  tarfile = archive_path,
  files = griis_file,
  tar = "internal"
)

unpacked <- bf_unpack_griis(
  archive_path = archive_path,
  exdir = file.path(archive_root, "unpacked"),
  quiet = TRUE
)

dir.exists(unpacked)
```

`bf_web_native_gbif` *Fetch native-range evidence from the GBIF Species API*

Description

Matches species names to GBIF taxon keys, retrieves checklist distribution records, keeps distribution rows with native-like establishment/status wording, and attempts to convert reported areas or country codes to ISO3.

Usage

```
bf_web_native_gbif(
  species,
  cache_dir = NULL,
  force_refresh = FALSE,
  sleep_sec = 0.25,
  user_agent = "biofetchR native-range web helper (academic use)",
  quiet = FALSE
)
```

Arguments

`species`	Character vector of species names.

cache_dir	Directory used to cache GBIF JSON responses. Must be supplied explicitly. In examples, tests and vignettes, use a path under tempdir().
force_refresh	Logical; if TRUE, ignore cached responses and fetch from the API again.
sleep_sec	Delay between requests, in seconds.
user_agent	HTTP user-agent string.
quiet	Logical; suppress progress messages.

Value

A tibble with one row per retained native-like GBIF Species API distribution record. The output uses the standard long native-evidence schema: species, the requested species name; source, set to "GBIF"; accepted_name, the matched GBIF scientific or canonical name; source_taxon_id, the GBIF taxon key; raw_native_area, the source country, area, locality or location string; raw_status, the establishment or status text used for native-like screening; origin_iso3, the resolved ISO3 country code where available; evidence_type, usually "native_distribution"; and source_url, the GBIF API endpoint used. If no native-like records are retained, an empty tibble with the same columns is returned.

Data source and interpretation

This helper uses GBIF Species API checklist-distribution records, not GBIF occurrence downloads. Retained rows are evidence of native-like distribution language in the API response and should be audited before being treated as a definitive native range.

See Also

Other native-range web-source helpers: bf_available_native_web_sources(), bf_fetch_native_ranges_web(), bf_web_native_worms(), bf_write_native_web_outputs()

Examples

```
if (interactive()) {
  gbif_native <- bf_web_native_gbif(
    species = "Carcinus maenas",
    cache_dir = file.path(tempdir(), "biofetchR_native_web"),
    quiet = FALSE
  )

  gbif_native
}
```

`bf_web_native_worms` *Fetch native-range evidence from WoRMS REST distributions*

Description

Resolves species names to valid AphiaIDs through the WoRMS REST API, retrieves Aphia distribution records, keeps native-like distribution/status rows, and attempts to map reported localities to ISO3 countries.

Usage

```
bf_web_native_worms(
  species,
  cache_dir = NULL,
  force_refresh = FALSE,
  sleep_sec = 0.25,
  marine_only = FALSE,
  user_agent = "biofetchR native-range web helper (academic use)",
  quiet = FALSE
)
```

Arguments

`species`	Character vector of species names.
`cache_dir`	Directory used to cache WoRMS JSON responses. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.
`force_refresh`	Logical; if `TRUE`, ignore cached responses and fetch from the API again.
`sleep_sec`	Delay between requests, in seconds.
`marine_only`	Logical passed to the WoRMS name-resolution endpoint.
`user_agent`	HTTP user-agent string.
`quiet`	Logical; suppress progress messages.

Value

A tibble with one row per retained native-like WoRMS Aphia distribution record. The output uses the standard long native-evidence schema: `species`, the requested species name; `source`, set to `"WoRMS"`; `accepted_name`, the valid or accepted WoRMS name; `source_taxon_id`, the AphiaID; `raw_native_area`, the source locality, area or geounit string; `raw_status`, the status, origin or establishment text used for native-like screening; `origin_iso3`, the resolved ISO3 country code where available; `evidence_type`, usually `"native_distribution"`; and `source_url`, the WoRMS REST endpoint used. If no native-like records are retained, an empty tibble with the same columns is returned.

Data source and interpretation

This helper uses WoRMS REST Aphia name and distribution endpoints. Retained rows are native-like distribution evidence and may remain unmapped when the source locality cannot be resolved to an ISO3 country code.

See Also

Other native-range web-source helpers: bf_available_native_web_sources(), bf_fetch_native_ranges_web(), bf_web_native_gbif(), bf_write_native_web_outputs()

Examples

```
if (interactive()) {
  worms_native <- bf_web_native_worms(
    species = "Carcinus maenas",
    cache_dir = file.path(tempdir(), "biofetchR_native_web"),
    quiet = FALSE
  )

  worms_native
}
```

---

bf_write_native_web_outputs

Write native-range web evidence outputs

Description

Writes the list returned by bf_fetch_native_ranges_web() to CSV files so the web-derived native-origin evidence can be audited and reused.

Usage

```
bf_write_native_web_outputs(x, output_dir, prefix = "native_web")
```

Arguments

x	Result from bf_fetch_native_ranges_web(return = "list").
output_dir	Directory for CSV outputs.
prefix	Filename prefix. Defaults to "native_web".

Value

Invisibly returns a named character vector of file paths written to output_dir. The vector contains paths named long, species, unmapped and summary, corresponding to the CSV files written from the matching elements of x. The function is called primarily for its side effect of writing auditable native-web evidence tables to disk.

Audit role

Writing these outputs preserves the exact web-derived evidence used by a pipeline run, including unmapped source strings. This is useful for later manual checking and for reporting which sources contributed native-origin evidence.

See Also

Other native-range web-source helpers: `bf_available_native_web_sources()`, `bf_fetch_native_ranges_web()`, `bf_web_native_gbif()`, `bf_web_native_worms()`

Examples

```
native_web <- list(
  long = data.frame(
    species = "Example species",
    source = "example",
    accepted_name = "Example species",
    source_taxon_id = "example_id",
    raw_native_area = "Exampleland",
    raw_status = "native",
    origin_iso3 = "GBR",
    evidence_type = "native_distribution",
    source_url = NA_character_
  ),
  species = data.frame(
    species = "Example species",
    native_origin_iso3 = "GBR",
    native_sources_used = "example",
    native_web_unmapped_strings = NA_character_,
    native_web_n_records = 1L,
    native_has_origin = TRUE
  ),
  unmapped = data.frame(),
  summary = data.frame(
    n_species_input = 1L,
    n_species_with_origin = 1L,
    n_records_long = 1L,
    n_records_unmapped = 0L
  )
)

out_dir <- tempfile("native_web_outputs_")
paths <- bf_write_native_web_outputs(native_web, output_dir = out_dir)
names(paths)
```

---

check_entrez_key *Check for NCBI Entrez API Key*

Description

Check for NCBI Entrez API Key

Usage

```
check_entrez_key()
```

Value

Invisibly returns a logical value of length one. The value is TRUE when the ENTREZ_KEY environment variable is present and non-empty, and FALSE otherwise. The function is called primarily for its side effect of reporting whether an NCBI Entrez API key is available for optional taxonomy-related workflows.

---

check_gbif_presence *Check whether GBIF has coordinate-based records for a species*

Description

Query the GBIF occurrence-search API through `rgbif::occ_search()` and return a simple TRUE/FALSE flag indicating whether at least one coordinate-bearing record is available for a species. The helper is intended as a lightweight pre-check before running heavier GBIF download workflows.

Usage

```
check_gbif_presence(
  species,
  region_id = NULL,
  use_eez_for_marine = FALSE,
  verbose = FALSE
)
```

Arguments

species	Character scalar. Scientific name to query.
region_id	Character scalar or NULL. ISO2 country code used for the GBIF country predicate. Use NULL for a global check.
use_eez_for_marine	Logical. If TRUE, skip the country predicate for species detected as marine by is_marine_species().
verbose	Logical. If TRUE, print a short CLI status message.

Details

By default, `region_id` is passed to GBIF as an ISO2 country filter. When `use_eez_for_marine = TRUE` and the package-level `is_marine_species()` helper identifies the species as marine, the country predicate is skipped so the query behaves as a global marine pre-check.

This function only checks record availability. It does not submit or import a GBIF occurrence download and therefore does not generate a download DOI.

Value

A logical value of length one. The value is `TRUE` when the GBIF occurrence-search query returns at least one coordinate-bearing record for the supplied species and region settings. The value is `FALSE` when no matching coordinate-bearing record is returned or when the query fails. This result is an availability pre-check only and does not represent a GBIF occurrence download or download DOI.

GBIF data use

This helper uses GBIF's occurrence-search API for a quick availability check. Records used in analyses should still be cited through an appropriate GBIF download DOI, derived dataset, or dataset-level citation workflow.

See Also

Other GBIF helpers: `get_taxon_key()`, `wait_and_import_gbif()`

Examples

```
if (interactive()) {
  has_records <- check_gbif_presence(
    species = "Carcinus maenas"
  )

  has_records
}
```

`download_gbif_batch` *Submit global GBIF download jobs for species-level workflows*

Description

Submit one GBIF occurrence download per species without applying a GBIF country predicate. This backend is primarily used by the marine pipeline, where occurrences are first retrieved globally and then assigned to marine overlays such as EEZs, LMEs, MEOW ecoregions, IHO seas, FAO regions, or other package-managed marine polygons.

Usage

```
download_gbif_batch(
  batch = NULL,
  user,
  pwd,
  email,
  species = NULL,
  predicate_extra = list(),
  quiet = FALSE,
  max_active_downloads = 1L,
  slot_poll_seconds = 30,
  slot_max_polls = 120L,
  submission_max_tries = 120L,
  ...
)
```

Arguments

batch	Optional data frame containing a species column. If supplied, unique non-empty species names are taken from this column.
user	GBIF username. Required for rgbif::occ_download().
pwd	GBIF password. Required for rgbif::occ_download().
email	GBIF account email. Required for rgbif::occ_download().
species	Optional character vector of species names. Used only when batch is not supplied or does not contain a species column.
predicate_extra	Optional list of additional rgbif predicates to append to each download request. Defaults to an empty list.
quiet	Logical. If TRUE, suppress progress messages.
max_active_downloads	Integer. Conservative maximum number of PREPARING/RUNNING GBIF downloads allowed for the authenticated account before a new request is submitted. Defaults to 1.
slot_poll_seconds	Numeric. Seconds between account-level download-slot checks and retry sleeps.
slot_max_polls	Integer. Maximum number of account-level slot checks before submission is attempted and server-side retry handling takes over.
submission_max_tries	Integer. Maximum number of retries when GBIF explicitly reports that download capacity is full.
...	Additional arguments reserved for compatibility with pipeline code and future backend extensions. Currently ignored.

Details

Each species name is resolved with `rgbif::name_backbone()` and submitted to GBIF using a conservative occurrence predicate requiring:

- the resolved GBIF `taxonKey`;
- `hasCoordinate = TRUE`;
- `hasGeospatialIssue = FALSE`;
- `occurrenceStatus = "PRESENT"`.

Additional GBIF predicates can be appended through `predicate_extra`, allowing callers to add date, basis-of-record, dataset, licence or other GBIF filters without changing the backend.

Value

A named character vector containing GBIF occurrence-download keys for successfully submitted species. Names are the corresponding species names. The vector can be empty or contain fewer elements than the number of species requested when one or more species cannot be resolved or submitted.

The returned vector carries a `submission_audit` attribute with one row per requested species and the columns `label`, `gbif_key`, `submission_status`, `message`, and `attempts`. Successful rows use `submission_status = "submitted"`; failed rows retain the reason and attempt count so that missing species cannot be silently confused with zero GBIF occurrences.

Queue-aware submission

Each species is submitted through the shared queue-aware submission helper. Before each request, biofetchR checks the authenticated account for `PREPARING`/`RUNNING` GBIF occurrence downloads. The default `max_active_downloads = 1L` therefore avoids intentionally creating multiple unfinished asynchronous downloads from this backend.

If GBIF nevertheless reports that download capacity is full, the same species request is retained and retried rather than being discarded.

Submission audit and partial results

The returned key vector carries a `submission_audit` attribute with one row per requested species and the columns `label`, `gbif_key`, `submission_status`, `message`, and `attempts`.

Species-level failures do not remove keys already obtained for other species. Consequently, the returned character vector can contain only the successfully submitted species while the audit records failures such as `taxon_key_failed`, `submit_no_key`, `submit_failed`, and `submit_timeout`.

GBIF data use and citation

This function submits live GBIF occurrence downloads and returns GBIF download keys. GBIF-mediated data are free to access, but users who use GBIF downloads in research, reporting or policy should cite the DOI generated for each GBIF download, or an appropriate derived dataset DOI where applicable. Individual records may also originate from datasets with their own licences and citation requirements. biofetchR does not redistribute GBIF data or replace GBIF's data-use and citation requirements.

Side effects

Submits live GBIF occurrence download jobs through `rgbif::occ_download()`. The function returns download keys only; it does not wait for, import, clean, thin, or spatially join downloaded occurrence records.

Errors and failure handling

The function stops for invalid function-level configuration, including no valid species names, missing GBIF credentials, or a non-list `predicate_extra`.

Failure to resolve or submit an individual species is handled fail-soft: the species receives an explicit row in the attached `submission_audit`, while processing can continue for other species. This prevents a later failed submission from discarding download keys that were successfully obtained earlier in the call.

See Also

`download_gbif_batch_gadm()`, `wait_and_import_gbif_safe()`

Other GBIF download backends: `download_gbif_batch_gadm()`, `wait_and_import_gbif_safe()`

Examples

```
gbif_env <- c("GBIF_USER", "GBIF_PWD", "GBIF_EMAIL")

if (interactive() && all(nzchar(Sys.getenv(gbif_env)))) {
  keys <- download_gbif_batch(
    species = c("Ficopomatus enigmaticus", "Carcinus maenas"),
    user = Sys.getenv("GBIF_USER"),
    pwd = Sys.getenv("GBIF_PWD"),
    email = Sys.getenv("GBIF_EMAIL")
  )
}
```

---

download_gbif_batch_gadm

Submit one multi-country GBIF download for a terrestrial/freshwater species

---

Description

`download_gbif_batch_gadm()` provides the queue-aware GBIF submission backend used by `process_gbif_terrestrial_`

Rather than creating one asynchronous GBIF download job for every species-country combination, all valid requested countries for the supplied species are combined into a single GBIF request. The resulting archive is subsequently imported once and split back to country-level occurrence objects by `wait_and_import_gbif()`.

This design reduces the number of unfinished asynchronous GBIF jobs created by multi-country workflows while preserving one explicit country-level request label, retrieval outcome and downstream output for every requested country.

Usage

```
download_gbif_batch_gadm(
  species,
  iso2_codes,
  user,
  pwd,
  email,
  max_active_downloads = 1L,
  slot_poll_seconds = 30,
  slot_max_polls = 120L,
  submission_max_tries = 120L
)
```

Arguments

species	Character scalar. Scientific name of the species to resolve and submit to GBIF. Only the first supplied value is used after trimming.
iso2_codes	Character vector of requested ISO2 country codes. Codes are trimmed, converted to upper case, deduplicated, and stripped of missing or empty values before submission.
user	GBIF username used for authenticated occurrence-download submission and account-level queue inspection.
pwd	GBIF password used for authenticated occurrence-download submission and account-level queue inspection.
email	GBIF account email passed to `rgbif::occ_download()`.
max_active_downloads	Integer. Conservative maximum number of PREPARING/RUNNING GBIF occurrence downloads permitted for the authenticated account before another request is attempted. Defaults to 1L.
slot_poll_seconds	Numeric. Seconds between proactive account-level download-slot checks and between retries following recognised GBIF capacity errors.
slot_max_polls	Integer. Maximum number of proactive account-level slot checks performed before submission is attempted and server-side retry handling becomes the final capacity guard.
submission_max_tries	Integer. Maximum number of attempts when GBIF repeatedly reports that asynchronous download capacity is full.

Details

Submit a single asynchronous GBIF occurrence download for one species across one or more requested ISO2 countries. This backend is used by the terrestrial/freshwater biofetchR pipeline, where

the logical input and audit units are species x recipient-country combinations but the GBIF download unit can contain several countries for the same species.

The function performs the following steps:

1. Input standardisation: `species` is trimmed to a single character value. `iso2_codes` are converted to upper case, trimmed, deduplicated and stripped of missing or empty values.
2. GBIF taxon resolution: The species is resolved to a GBIF taxon key. When the package-visible `get_taxon_key()` helper is available it is used; otherwise the function falls back to `rgbif::name_backbone()`.
3. Multi-country predicate construction: All requested ISO2 countries are combined into one `rgbif::pred_in()` country predicate. The submitted occurrence-download request therefore contains:
   - the resolved GBIF `taxonKey`;
   - `hasCoordinate = TRUE`;
   - `country IN <all requested ISO2 country codes>`.
4. Queue-aware submission: The request is submitted through biofetchR's queue-aware retry helper. Before each submission attempt, the authenticated GBIF account is checked for currently `PREPARING` or `RUNNING` occurrence downloads.
5. Capacity-limit recovery: If GBIF explicitly reports that the account has reached its current incomplete/concurrent-download capacity, the SAME multi-country request is retained, paused and retried. Countries are not dropped merely because a submission attempt encountered a temporary GBIF capacity limit.
6. Shared-key reconstruction metadata: After a successful submission, every requested ISO2 country is mapped to the same GBIF download key. The returned object carries metadata telling `wait_and_import_gbif()` that this is a shared multi-country download and that the imported archive should be split using GBIF's `countryCode` column.

Value

On successful submission, a named list with one element for every requested ISO2 country. Each element contains the SAME shared GBIF occurrence-download key because all countries were submitted in one multi-country request.

The returned object carries a `submission_audit` attribute and, for the multi-country workflow, the attributes `biofetchR_shared_download`, `biofetchR_split_field`, and `biofetchR_requested_labels`. These attributes allow `wait_and_import_gbif()` to poll/import the shared archive once and reconstruct country-level results.

If taxon resolution or GBIF submission fails for otherwise valid input, a length-zero object can be returned with submission-audit information describing the failure. Invalid or empty species/country input can return `NULL` after a warning.

Species-country requests versus GBIF download jobs

A central distinction in this backend is that the biological/request unit and the asynchronous GBIF job unit are not necessarily the same.

For example, an input consisting of:

- species A x country GB;
- species A x country IE;
- species A x country FR;
- species A x country ES

remains four species-country retrieval requests for auditing and downstream processing, but is submitted to GBIF as ONE asynchronous occurrence download for species A with:
`country IN c("GB", "IE", "FR", "ES")`.

The completed GBIF archive is then imported once and split locally back into the four requested country subsets.

Queue and concurrency protection

GBIF occurrence downloads are asynchronous and the number of incomplete jobs allowed for one account can be limited by the service. The backend therefore applies conservative queue protection before submission.

`max_active_downloads` controls the maximum number of account-level `PREPARING`/`RUNNING` downloads permitted before another request is attempted. The default is `1L`, meaning that biofetchR waits while another active download is visible before submitting a new one.

Account-level queue checks are repeated every `slot_poll_seconds` seconds for at most `slot_max_polls` checks. If the account-level download list cannot be inspected reliably, submission is allowed to proceed and GBIF's server-side response becomes the final capacity guard.

If GBIF then returns a recognised concurrency or incomplete-download limit, the same request is retried up to `submission_max_tries` times. A temporary capacity error therefore does not consume a country request or silently remove that country from the workflow.

Shared GBIF download keys

On successful submission, the returned object contains one named list element for every requested ISO2 country. All elements intentionally contain the SAME GBIF download key because all countries were included in one asynchronous download.

For example, the returned object may conceptually contain:

```
GB = "0001234-260101120000000"
IE = "0001234-260101120000000"
FR = "0001234-260101120000000"
ES = "0001234-260101120000000"
```

This repeated key is expected behaviour and preserves the pre-0.1.1 country-labelled list interface while avoiding multiple redundant GBIF jobs.

`wait_and_import_gbif()` deduplicates the shared key, polls and imports the archive once, then reconstructs separate country-labelled results using `countryCode`.

Returned metadata attributes

The returned object can carry the following attributes:

- `submission_audit`: submission-level audit information;
- `biofetchR_shared_download = TRUE`: identifies the object as a shared multi-country GBIF download;
- `biofetchR_split_field = "countryCode"`: tells the importer which GBIF field should be used to reconstruct country subsets;
- `biofetchR_requested_labels`: the original requested ISO2 country labels.

These attributes are internal workflow metadata and should normally be passed unchanged to `wait_and_import_gbif()` or `wait_and_import_gbif_safe()`.

Submission audit

The `submission_audit` attribute contains one row per requested ISO2 country, even though the countries share one GBIF job. Its fields are:

- `label`: requested ISO2 country code;
- `gbif_key`: shared GBIF download key when available;
- `submission_status`: submission outcome;
- `message`: diagnostic message when relevant;
- `attempts`: number of GBIF submission attempts used.

Successful countries normally share `submission_status = "submitted"` and the same `gbif_key`. Failed taxon resolution or download submission is represented explicitly in the audit rather than being silently interpreted as a zero-occurrence result.

Failure handling

Empty or invalid species/country input is rejected before submission.

If no GBIF taxon key can be resolved for an otherwise valid species request, the function returns a length-zero object carrying a submission audit with `submission_status = "taxon_key_failed"`.

Submission-level failures are returned through statuses produced by the queue-aware submission helper, including:

- `submit_no_key`: GBIF returned a response but no usable download key could be extracted;
- `submit_failed`: submission failed for a non-capacity reason;
- `submit_timeout`: the configured capacity-retry limit was exhausted.

These states remain distinct from downstream polling/import failures handled by `wait_and_import_gbif()`. In particular, a failed submission or unresolved asynchronous download must not be interpreted as evidence that GBIF contained zero occurrence records for the country.

GBIF data use and citation

This function submits live GBIF occurrence downloads and returns GBIF download keys. Users should retain those keys and cite the corresponding GBIF download DOI(s) for occurrence records used in research, reports or policy outputs.

A single multi-country submission normally corresponds to one GBIF download DOI shared by all country subsets reconstructed from that archive. The country-specific outputs should therefore retain the common GBIF key so that all derived records remain traceable to the exact source download.

GBIF aggregates occurrence records from many contributing datasets. Individual datasets may carry their own licences and attribution requirements. biofetchR does not redistribute GBIF occurrence data and does not replace users' responsibility to comply with GBIF and publisher data-use terms.

Side effects

Submits a live asynchronous GBIF occurrence-download request through `rgbif::occ_download()` when a usable taxon key and at least one ISO2 country are available.

The function can also query the authenticated user's GBIF download list through `rgbif::occ_download_list()` as part of proactive queue management, and prints submission, waiting and retry messages through `cli`.

The function itself does not wait for the submitted occurrence download to finish, download the SIMPLE_CSV archive, clean coordinates, thin occurrences, or assign records to spatial units. Those operations occur downstream.

Notes

The backend preserves the historical public return shape of `download_gbif_batch_gadm()`: a named list whose names represent requested countries.

The important behavioural change is that those country labels can now point to one shared GBIF key. Code should therefore not assume that the number of returned list elements equals the number of distinct GBIF download jobs.

See Also

`download_gbif_batch()`, `wait_and_import_gbif()`, `wait_and_import_gbif_safe()`

Other GBIF download backends: `download_gbif_batch()`, `wait_and_import_gbif_safe()`

Examples

```
gbif_env <- c("GBIF_USER", "GBIF_PWD", "GBIF_EMAIL")

if (interactive() && all(nzchar(Sys.getenv(gbif_env)))) {
  keys <- download_gbif_batch_gadm(
    species = "Hydrocotyle ranunculoides",
    iso2_codes = c("GB", "IE"),
    user = Sys.getenv("GBIF_USER"),
    pwd = Sys.getenv("GBIF_PWD"),
    email = Sys.getenv("GBIF_EMAIL")
  )
```

```
  keys
  attr(keys, "submission_audit")
}
```

---

eez_join *Join GBIF points to Exclusive Economic Zone polygons*

Description

Loads or caches Exclusive Economic Zone (EEZ) polygons via **mregions2**, repairs invalid geometries where possible, harmonises common EEZ attribute names across Marine Regions releases, and joins each occurrence-point `sf` object in `results_list` to an EEZ polygon.

Usage

```
eez_join(results_list, use_planar = TRUE, eez_cache_file = NULL)
```

Arguments

`results_list`	Named list of point `sf` objects. Each list element is usually one species or one species-country download result. A single `sf` object is also accepted and will be wrapped into a length-one list.
`use_planar`	Logical. If `TRUE`, disable spherical geometry operations with `sf::sf_use_s2(FALSE)` and run joins in a projected CRS. If `FALSE`, the function first tries the layer CRS and falls back to planar joins when spherical operations fail.
`eez_cache_file`	Path to a GeoPackage used to cache the EEZ layer. Must be supplied explicitly. In examples, tests and vignettes, use a path under `tempdir()`.

Details

The function returns the same named list structure as the input and appends standardised EEZ metadata used by downstream native-status and marine-overlay workflows. When a point does not fall strictly within an EEZ polygon, the function attempts a nearest-feature fallback so that points lying close to polygon boundaries can still be assigned cautiously.

EEZ source attributes differ across Marine Regions releases. This helper uses case-insensitive matching to find common territory, sovereign and ISO-code fields, then derives best-effort ISO3 and ISO2 codes for downstream matching. If **countrycode** is installed, missing ISO codes are inferred from cleaned territory, sovereign or EEZ names where possible.

Value

A named list with the same names as `results_list`. Each element is an `sf` point object containing the original occurrence records plus standardised Exclusive Economic Zone attributes where a spatial match could be made. Added columns include `geoname`, `mrgid`, `territory1`, `sovereign1`, `iso3_best` and `iso2_best`. These columns identify the matched marine region and provide best-effort country codes for downstream filtering, summarisation and export. Records that cannot be assigned retain the input geometry and receive missing values in the added fields.

Data access and attribution

EEZ polygons are retrieved from Marine Regions through **mregions2** and cached locally. biofetchR does not redistribute Marine Regions data. Users should cite Marine Regions and check the applicable data licence and attribution requirements for the version used in their analysis.

See Also

Other marine overlay joins: `overlay_join()`

Examples

```
if (interactive()) {
  pts <- data.frame(
    decimalLongitude = c(-5, 10),
    decimalLatitude = c(50, 35)
  )

  pts_sf <- sf::st_as_sf(
    pts,
    coords = c("decimalLongitude", "decimalLatitude"),
    crs = 4326,
    remove = FALSE
  )

  joined <- eez_join(
    list("Carcinus maenas" = pts_sf),
    eez_cache_file = file.path(tempdir(), "biofetchR_eez_cache.gpkg")
  )

  names(joined[["Carcinus maenas"]])
}
```

`filter_by_status` *Filter or normalise records by native/alien status*

Description

Normalise a provider-specific origin, establishment or alien-status column to biofetchR's canonical status classes and optionally filter rows to a supported preset.

Usage

```
filter_by_status(
  data,
  status_col = "alien_status",
  preset = "all",
  values = NULL,
  normalize_only = FALSE,
  verbose = TRUE
)
```

Arguments

data	A data.frame or tibble containing status information.
status_col	Character scalar. Column containing raw status labels. Defaults to "alien_status".
preset	Character scalar. One of list_status_presets()$preset. Defaults to "all", which retains all canonical classes.
values	Optional character vector of canonical statuses to keep, such as c("introduced", "naturalized"). When supplied, this overrides preset.
normalize_only	Logical. If TRUE, add status_norm but do not filter rows.
verbose	Logical. If TRUE, print a short retained-row summary.

Details

The function adds or updates a status_norm column using the internal status mapper. When normalize_only = TRUE, no rows are removed. Otherwise, rows are retained when status_norm belongs to the classes resolved from preset or values.

This helper is deliberately tolerant of missing status columns. If status_col is absent, the input data are returned unchanged. A warning is emitted only when a real filtering request was made.

Value

A data frame or tibble of the same general type as data, with a status_norm column added or updated when status_col is present. status_norm contains canonical biofetchR status classes such as "native", "introduced", "naturalized", "invasive", "cryptogenic", "managed" or "unknown". If normalize_only = TRUE, all input rows are returned with the normalised status column. Otherwise, rows are filtered to the classes selected by preset or values. If status_col is absent, the input data are returned unchanged.

See Also

Other status helpers: list_status_presets()

Examples

```
x <- data.frame(
  species = c("Species a", "Species b", "Species c"),
  alien_status = c("native", "introduced", "cryptogenic")
```

```
)

filter_by_status(x, preset = "non_native", verbose = FALSE)
filter_by_status(x, normalize_only = TRUE, verbose = FALSE)
```

---

gadm_join *Attach GADM labels to occurrence points*

Description

Join point records to GADM polygons and append one set of GADM identifier/name columns to each point. The join is intentionally one-to-one: if a point intersects multiple polygons, only the first polygon hit is used, which avoids accidental row multiplication during GBIF export.

Usage

```
gadm_join(
  pts_sf,
  gadm,
  level = 1,
  id_out = "id_gadm",
  name_out = "name_gadm",
  geoname_out = "geoname_gadm",
  quiet = TRUE
)
```

Arguments

pts_sf	sf point object. EPSG:4326 is recommended; other CRSs are transformed to WGS84 where possible.
gadm	Either a named list from load_gadm() or a single sf polygon layer containing GADM geometries.
level	Integer GADM level used to select gid_<level> and name_<level> columns.
id_out	Name of the output GADM identifier column. Defaults to "id_gadm".
name_out	Name of the output GADM name column. Defaults to "name_gadm".
geoname_out	Name of the output human-readable GADM label column. Defaults to "geoname_gadm".
quiet	Logical; suppress warning/progress messages when TRUE.

Details

gadm_join() accepts either the named list returned by load_gadm() or a single pre-bound GADM sf polygon layer. Both points and polygons are transformed to WGS84 where possible before the intersection test. When no valid polygon layer is available, the output columns are still created and filled with NA_character_, keeping the downstream schema stable.

Value

An sf point object with the same rows as the input pts_sf and with three GADM assignment columns appended. The column named by id_out contains the matched GADM identifier, the column named by name_out contains the matched administrative-unit name, and the column named by geoname_out contains the human-readable GADM label used by downstream exports. Points with no polygon match receive NA_character_ in the added columns. The function performs a one-to-one assignment and does not duplicate point rows when polygons overlap.

See Also

Other GADM spatial helpers: load_all_gadm(), load_gadm()

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  pts <- sf::st_as_sf(
    data.frame(
      species = "Example species",
      decimalLongitude = -5.93,
      decimalLatitude = 54.60
    ),
    coords = c("decimalLongitude", "decimalLatitude"),
    crs = 4326,
    remove = FALSE
  )

  poly <- sf::st_sf(
    gid_1 = "GBR.1_1",
    name_1 = "Example region",
    GID = "GBR.1_1",
    GADM_name = "Example region",
    geometry = sf::st_sfc(
      sf::st_polygon(list(rbind(
        c(-7, 53),
        c(-4, 53),
        c(-4, 56),
        c(-7, 56),
        c(-7, 53)
      ))),
      crs = 4326
    )
  )

  joined <- gadm_join(pts, poly, level = 1)
  names(joined)
}
```

get_taxon_key *Retrieve a GBIF taxonKey for a scientific name*

Description

Resolve a supplied name against the GBIF backbone taxonomy using `rgbif::name_backbone()` and return the matched `usageKey` when available.

Usage

```
get_taxon_key(species, verbose = FALSE)
```

Arguments

`species`	Character scalar. Scientific name to resolve.
`verbose`	Logical. If `TRUE`, print a CLI status message.

Details

This helper is a thin, fail-soft wrapper around `rgbif::name_backbone()`. It returns `NA_integer_` rather than stopping when GBIF name matching fails, which makes it suitable for batch pipelines and audit tables.

Value

An integer value of length one containing the matched GBIF backbone `usageKey` for `species`. Returns `NA_integer_` when no usage key is found or when the GBIF backbone lookup fails. The value can be used as a GBIF taxon identifier in downstream occurrence-download predicates and audit tables.

See Also

Other GBIF helpers: `check_gbif_presence()`, `wait_and_import_gbif()`

Examples

```
if (interactive()) {
  key <- get_taxon_key("Carcinus maenas")

  key
}
```

`initialize_summary` *Initialise a GBIF processing summary table*

Description

Create an empty, typed summary table for tracking the outcome of GBIF occurrence processing across species and spatial recipient units.

Usage

```
initialize_summary()
```

Details

`initialize_summary()` is used by biofetchR pipelines before records are downloaded, imported, cleaned, thinned and exported. The returned table has a stable schema that can be extended by higher-level workflows when additional audit fields are needed.

The summary table is designed around one row per processed species x region combination. The `region_id` and `region_type` fields are intentionally generic so the same helper can be used for terrestrial/freshwater workflows based on GADM units and for marine workflows based on EEZs or other marine overlays.

The base columns are:

`species` Scientific name or accepted taxon label used by the pipeline.

`region_id` Spatial recipient identifier, such as a GADM code, EEZ label or marine-region identifier.

`region_type` Spatial-recipient family, such as `"GADM"`, `"EEZ"` or another overlay label.

`n_total` Number of records before coordinate cleaning.

`n_cleaned` Number of records remaining after coordinate cleaning.

`n_thinned` Number of records remaining after spatial thinning.

`output_file` Path to the exported CSV file, or `NA` when records are retained only in memory.

`status` Processing status, such as `"success"`, `"empty"` or `"failed"`.

Value

A zero-row `tibble::tibble()` with typed columns used to track GBIF processing outcomes. The returned table contains character columns `species`, `region_id`, `region_type`, `output_file` and `status`, and integer columns `n_total`, `n_cleaned` and `n_thinned`. Each future row is intended to represent one processed species x spatial-recipient combination, with counts for records before cleaning, after coordinate cleaning and after spatial thinning. The empty table is used as the starting summary object for terrestrial/freshwater and marine processing pipelines.

See Also

Other processing summary helpers: `append_summary_row()`

Examples

```
summary_tbl <- initialize_summary()
summary_tbl
```

`install_optional_deps` *Report optional biofetchR dependencies*

Description

Reports whether optional packages used by extended biofetchR workflows are available. This function reports package availability only. Optional packages should be installed by the user outside package examples, tests and vignettes.

Usage

```
install_optional_deps()
```

Value

A data frame with one row per optional package and columns: `package`, `available` and `purpose`.

Examples

```
install_optional_deps()
```

`is_marine_species` *Determine if a species is likely marine from taxonomy*

Description

Uses GBIF taxonomy first. If needed, falls back to broad marine clades and a small genus-name heuristic.

Usage

```
is_marine_species(species_name, verbose = FALSE)
```

Arguments

`species_name`	Scientific name.
`verbose`	Print result.

Value

A logical value of length one. The value is TRUE when the supplied species name is identified as likely marine from resolved taxonomy clades or a small genus-name heuristic, and FALSE otherwise. When verbose = TRUE, the function also reports the classification result as a message.

---

list_status_presets *List available native/non-native status presets*

Description

Return the status-filter presets recognised by filter_by_status(). These presets define how raw origin, establishment or alien-status labels are grouped into analysis-ready categories such as native, introduced, naturalized, invasive or cryptogenic.

Usage

```
list_status_presets()
```

Details

biofetchR uses these presets when users want consistent filtering across provider-specific status fields. The presets are intentionally conservative: "non_native" and its alias "alien" exclude cryptogenic records unless the explicit "non_native_plus_cryptogenic" preset is requested.

Value

A base data.frame describing the status-filter presets recognised by filter_by_status(). The table contains three character columns: preset, the preset name accepted by filter_by_status(); includes, the canonical status classes retained by that preset; and notes, a short interpretation of the preset. Each row represents one supported preset or alias for filtering native, non-native, invasive, cryptogenic, managed or unknown status records.

See Also

Other status helpers: filter_by_status()

Examples

```
list_status_presets()
```

load_all_gadm *Load and bind GADM geometries for multiple countries*

Description

Convenience wrapper around `load_gadm()` that loads one or more countries and binds the returned list into a single `sf` object. A new `iso2c` column records the source country for each bound polygon.

Usage

```
load_all_gadm(
  iso_codes,
  level = 1,
  path = NULL,
  quiet = FALSE,
  force_refresh = FALSE
)
```

Arguments

`iso_codes`	Character vector of ISO2 country codes.
`level`	Integer GADM administrative level.
`path`	Cache directory passed to `load_gadm()` as `cache_dir`. If `NULL`, `load_gadm()` uses a package cache directory. In examples, tests and vignettes, supply a path under `tempdir()`.
`quiet`	Logical; suppress progress messages when `TRUE`.
`force_refresh`	Logical; re-download/rebuild cached GADM objects when `TRUE`.

Value

An `sf` polygon object containing all successfully loaded GADM countries bound into one layer. The returned object includes an `iso2c` column identifying the source country for each polygon, along with the GADM identifier and name columns standardised by `load_gadm()`, including `GID`, `GADM_name`, `gid_<level>` and `name_<level>`. If no country can be loaded, the function returns an empty `sf` object.

See Also

Other GADM spatial helpers: `gadm_join()`, `load_gadm()`

Examples

```
if (interactive()) {
  gadm_l1 <- load_all_gadm(
    iso_codes = c("GB", "IE"),
    level = 1,
    path = file.path(tempdir(), "biofetchR_gadm_cache")
```

```
  )

  names(gadm_l1)
}
```

`load_gadm` *Load GADM administrative geometries*

Description

Download, cache and standardise GADM administrative boundary polygons for one or more ISO2 country codes. The function returns a named list of `sf` objects, keyed by ISO2 code, with stable GADM identifier and name columns that can be used by downstream joins.

Usage

```
load_gadm(
  iso2c,
  level = 1,
  cache_dir = NULL,
  quiet = FALSE,
  force_refresh = FALSE
)
```

Arguments

`iso2c`	Character vector of ISO2 country codes, for example `"GB"` or `"IE"`. The common alias `"UK"` is converted to `"GB"`, and the literal ISO2 code `"NA"` is treated as Namibia.
`level`	Integer GADM administrative level to load, usually `0`, `1` or `2` in biofetchR workflows.
`cache_dir`	Character. Directory used for GADM downloads and cached objects. Must be supplied explicitly. In examples, tests and vignettes, use `file.path(tempdir(), ...)`.
`quiet`	Logical; suppress progress messages when `TRUE`.
`force_refresh`	Logical; re-download/rebuild the cleaned cache even when a cached RDS file exists.

Details

`load_gadm()` is designed to be robust across first-run downloads, cached RDS objects and schema differences in GADM/geodata outputs. It converts ISO2 codes to ISO3 codes, downloads boundaries through `geodata::gadm()` when no valid cache exists, and falls back to the official GADM 4.1 per-country shapefile archive when `geodata::gadm()` fails. The resulting object is converted to

sf, repaired where possible, transformed to WGS84, and normalised to the key columns expected by the terrestrial/freshwater pipeline.

The returned layers always include generic GID and GADM_name columns, plus level-specific gid_<level> and name_<level> columns. This makes downstream code less sensitive to whether the source object contains full GADM schemas or a compact packed schema.

Value

A named list of sf polygon objects, keyed by ISO2 country code. Each list element contains the successfully loaded GADM polygons for one country at the requested administrative level, transformed to WGS84 where possible. Returned layers include generic GID and GADM_name columns plus level-specific gid_<level> and name_<level> columns used by downstream joins. Invalid ISO2 codes or countries that cannot be loaded are skipped, so the returned list may be shorter than the input iso2c vector or empty. The function is also called for the side effect of downloading, repairing and caching GADM boundary data locally.

Data access and attribution

This function can download GADM boundary data through geodata. biofetchR does not ship or redistribute GADM data from this script. Users are responsible for checking the GADM licence, citation and redistribution terms for the boundary version used in their analyses.

See Also

Other GADM spatial helpers: gadm_join(), load_all_gadm()

Examples

```
if (interactive()) {
  gadm <- load_gadm(
    iso2c = c("GB", "IE"),
    level = 1,
    cache_dir = file.path(tempdir(), "biofetchR_gadm_cache")
  )

  names(gadm)
}
```

---

overlay_join *Join GBIF points to Marine Regions overlays (robust, with s2 fallback)*

Description

Reads/caches a Marine Regions polygon layer (EEZ/LME/IHO/MEOW/FAO, etc.), fixes invalid geometries, harmonizes name/id columns, and joins each sf in results_list to the polygons. If any s2 error occurs, the join transparently falls back to planar (sf_use_s2(FALSE)).

Usage

```
overlay_join(
  results_list,
  overlay = c("eez", "lme", "iho", "meow", "fao", "custom"),
  overlay_layer = NULL,
  overlay_sf = NULL,
  use_planar = TRUE,
  cache_dir = NULL
)
```

Arguments

`results_list`	Named list of sf point data (e.g., species -> sf).
`overlay`	Character; which overlay family to use. One of c("eez","lme","iho","meow","fao","custom"). Only affects defaults if `overlay_layer` is not supplied.
`overlay_layer`	Character; exact layer id for Marine Regions. If given, it is used directly. If NULL, a canonical id is chosen for the `overlay`. Examples: "eez", "lme", "iho", "meow", "fao".
`overlay_sf`	Optional `sf` polygon object (preloaded overlay). If provided, no download/caching is attempted.
`use_planar`	Logical; if TRUE, run joins with s2 disabled from the start (planar ops). Regardless, the function auto-falls back to planar if a spherical join fails.
`cache_dir`	Directory to cache the overlay as a GeoPackage. Must be supplied explicitly when `overlay_sf = NULL`. In examples, tests and vignettes, use a path under `tempdir()`.

Details

- Uses **mregions2** through `mrp_list`/`mrp_get`. Legacy Marine Regions fallback support has been removed.
- For non-EEZ overlays, `iso3_best`/`iso2_best` will likely be `NA`; your pipeline's `ensure_iso3_on_points()` will still fill ISO by overlaying a world-country layer later.

Value

A named list with the same names as `results_list`. Each element is an `sf` point object containing the original occurrence records plus standardised attributes from the matched Marine Regions overlay. The returned columns may include `geoname`, `mrgid`, `territory1`, `sovereign1`, `iso3_best` and `iso2_best`, depending on the overlay used. `geoname` is the matched marine-region label, `mrgid` is the Marine Regions identifier when available, and `iso3_best`/`iso2_best` are best-effort ISO country codes mainly available for Exclusive Economic Zone overlays. Records that cannot be assigned retain their input geometry and receive missing values in the added fields.

Data access and attribution

Marine Regions overlays are accessed through **mregions2** and cached locally. biofetchR does not redistribute these layers. Users should cite Marine Regions and check the data provider's licence and attribution requirements for the specific overlay used.

See Also

Other marine overlay joins: eez_join()

Examples

```
if (interactive()) {
  pts <- data.frame(
    decimalLongitude = c(-5, 10),
    decimalLatitude = c(50, 35)
  )

  pts_sf <- sf::st_as_sf(
    pts,
    coords = c("decimalLongitude", "decimalLatitude"),
    crs = 4326,
    remove = FALSE
  )

  joined <- overlay_join(
    list("Carcinus maenas" = pts_sf),
    overlay = "eez",
    cache_dir = file.path(tempdir(), "biofetchR_overlay_cache")
  )

  names(joined[["Carcinus maenas"]])
}
```

---

process_gbif_eez_pipeline

Run the marine pipeline through the legacy EEZ wrapper

Description

Compatibility wrapper for older code that still calls process_gbif_eez_pipeline(). All arguments are forwarded unchanged to process_gbif_marine_pipeline(). The wrapper is retained so existing scripts can continue to run while the package uses the more general marine-overlay workflow internally.

Usage

```
process_gbif_eez_pipeline(...)
```

Arguments

...	Arguments passed directly to process_gbif_marine_pipeline().

Value

Returns the same object as process_gbif_marine_pipeline(). When return_all_results = TRUE and store_in_memory = TRUE, this is a tibble containing combined processed marine occurrence records with marine-region assignment, coordinate, workflow and optional evidence-audit columns. Otherwise, the wrapper writes outputs through process_gbif_marine_pipeline() and returns invisible(NULL). This function is a backwards-compatible wrapper and is called primarily for the side effects of running the marine GBIF pipeline through the legacy EEZ interface.

See Also

Other marine processing pipelines: process_gbif_marine_pipeline()

---

process_gbif_marine_pipeline *Process marine GBIF occurrences using package-managed marine overlays*

Description

process_gbif_marine_pipeline() is the main marine processing workflow in biofetchR. It downloads/imports GBIF occurrence records for marine species, optionally applies pre-download evidence gates, assigns records to a selected marine spatial overlay, optionally cleans and thins the occurrence records, and exports per species x marine-region occurrence tables.

Usage

```
process_gbif_marine_pipeline(
  df,
  output_dir,
  user = Sys.getenv("GBIF_USER"),
  pwd = Sys.getenv("GBIF_PWD"),
  email = Sys.getenv("GBIF_EMAIL"),
  batch_size = 5,
  apply_cleaning = TRUE,
  apply_thinning = FALSE,
  dist_km = 5,
  return_all_results = TRUE,
  export_summary = TRUE,
  store_in_memory = TRUE,
  use_planar = TRUE,
  add_status = FALSE,
  overlay = "eez",
  overlay_sf = NULL,
  overlay_cache_dir = NULL,
  overlay_force_refresh = FALSE,
  strict_overlay_loading = TRUE,
  prepare_taxonomy = FALSE,
```

```
  manual_taxonomy_fixes = NULL,
  taxonomy_name_col = "species",
  export_taxonomy_audit = TRUE,
  use_griis_status = FALSE,
  griis = NULL,
  griis_filter_mode = c("audit_only", "listed", "invasive"),
  export_griis_audit = TRUE,
  use_native_status = FALSE,
  native_ranges = NULL,
  use_native_web = FALSE,
  native_web_sources = c("sinas", "gbif", "worms"),
  native_web_cache_dir = NULL,
  native_web_force_refresh = FALSE,
  native_web_sleep_sec = 0.25,
  native_web_sinas_main_path = NULL,
  native_web_sinas_alllocations_path = NULL,
  native_web_sinas_fulltaxa_path = NULL,
  export_native_web_audit = TRUE,
  native_species_col = "species",
  native_filter_mode = c("audit_only", "origin_available_only"),
  reconcile_griis_native = TRUE,
  export_native_audit = TRUE,
  quiet = FALSE,
  deps = NULL
)
```

Arguments

df	Data frame containing at least a species column. When prepare_taxonomy = TRUE, names are cleaned before GBIF submission and the cleaned accepted names are used downstream.
output_dir	Directory for per-region CSV exports, gbif_summary.csv, gbif_retrieval_audit.csv, gbif_unresolved_requests.csv, and optional taxonomy/GRIIS/native-origin audit files.
user	GBIF username.
pwd	GBIF password.
email	GBIF account email address.
batch_size	Number of species grouped into each outer processing batch. GBIF submission and import are performed species-by-species within each batch so this value no longer determines the number of concurrent GBIF jobs.
apply_cleaning	Logical; if TRUE, apply coordinate/quality cleaning through thin_spatial_points(..., dist_km = 0) when available.
apply_thinning	Logical; if TRUE, spatially thin records after overlay assignment using thin_spatial_points().
dist_km	Numeric thinning distance in kilometres.
return_all_results	Logical; if TRUE and store_in_memory = TRUE, return a combined table of processed occurrence rows.

export_summary Logical; if TRUE, write gbif_summary.csv, gbif_retrieval_audit.csv, and gbif_unresolved_requests.csv.

store_in_memory
Logical; if TRUE, retain processed groups in memory; if FALSE, write groups to CSV and return invisibly.

use_planar Logical; if TRUE, disable spherical S2 predicates during this run for compatibility with planar overlay operations.

add_status Logical retained for backwards compatibility. When TRUE, an empty occurrenceStatus column is created before cleaning if absent.

overlay Marine overlay name or alias, resolved by bf_marine_overlay_sources() / bf_load_marine_regions_overlay().

overlay_sf Optional preloaded sf polygon layer. Normally NULL, in which case the package-managed overlay loader is used.

overlay_cache_dir
Cache directory for package-managed marine overlay downloads and processed overlay objects. If NULL, a cache folder is created under the explicitly supplied output_dir.

overlay_force_refresh
Logical; rebuild or re-download overlay resources where supported.

strict_overlay_loading
Logical; if TRUE, stop when the requested overlay cannot be loaded.

prepare_taxonomy
Logical; if TRUE, run the taxonomy gate before GBIF download submission.

manual_taxonomy_fixes
Optional named character vector or data frame of manual taxonomic corrections applied before name cleaning.

taxonomy_name_col
Name of the input column containing taxon names.

export_taxonomy_audit
Logical; if TRUE, write taxonomy_audit.csv, taxonomy_rejected.csv and taxonomy_summary.csv when taxonomy preparation is enabled.

use_griis_status
Logical; if TRUE, attach species-level GRIIS status before GBIF submission.

griis Optional GRIIS table from bf_read_griis() or bf_standardise_griis().

griis_filter_mode
Character; one of "audit_only", "listed" or "invasive".

export_griis_audit
Logical; if TRUE, write griis_status_audit.csv.

use_native_status
Logical; if TRUE, attach species-level native-origin evidence before GBIF submission.

native_ranges Native-origin table or lookup accepted by bf_attach_native_status(). Leave NULL when use_native_web = TRUE so the pipeline can build species-level origin evidence from web/API sources after taxonomy and GRIIS filtering.

use_native_web Logical; if TRUE, fetch species-level native-origin evidence from package web/API helpers before applying the marine native evidence gate. Requires use_native_status = TRUE and native_ranges = NULL.

native_web_sources
Character vector of native web sources passed to bf_fetch_native_ranges_web(), commonly c("sinas", "gbif", "worms").

native_web_cache_dir
Cache directory for native web/API responses. If NULL, a cache folder is created under the explicitly supplied output_dir.

native_web_force_refresh
Logical; if TRUE, refresh native-web cache entries where supported.

native_web_sleep_sec
Numeric delay in seconds between native-web requests.

native_web_sinas_main_path
Optional local path to SInAS_3.1.1.csv when "sinas" is included in native_web_sources.

native_web_sinas_alllocations_path
Optional local path to AllLocations.xlsx, .csv or .tsv.

native_web_sinas_fulltaxa_path
Optional local path to SInAS_3.1.1_FullTaxaList.csv.

export_native_web_audit
Logical; if TRUE, write native-web long, species, unmapped and summary audit files to output_dir.

native_species_col
Species column in native_ranges when raw native range tables are supplied.

native_filter_mode
Character; one of "audit_only" or "origin_available_only".

reconcile_griis_native
Logical; if TRUE, add reconciled evidence status when both GRIIS and native-origin columns are present.

export_native_audit
Logical; if TRUE, write native_status_audit.csv.

quiet Logical; suppress console progress messages.

deps Optional named list of dependency overrides for tests. Supported entries include initialize_summary, append_summary_row, download_fun, import_fun and thin_fun.

Details

The pipeline is deliberately ordered so that problematic or unsupported rows are removed before live GBIF downloads are submitted:

1. The input species table is validated and deduplicated.
2. If requested, the taxonomy gate applies manual fixes, cleans names and rejects non-taxa, open names and higher-rank placeholders.
3. If requested, the GRIIS gate attaches species-level invasive-status evidence and can filter to listed or invasive species.

4. If requested, the native-origin gate attaches species-level origin evidence and can filter to species with origin evidence.
5. **Outer batching.** Retained species are grouped according to `batch_size` for progress and processing organisation.
6. **Sequential GBIF retrieval within each batch.** One species is submitted to the configured GBIF backend, polled and imported completely before the next species in that same outer batch is submitted.
7. **Retrieval reconciliation.** Each species receives an explicit GBIF retrieval outcome before downstream marine processing begins.
8. **Point and overlay preparation.** Successful imports are converted to WGS84 point `sf` objects and a package-managed marine overlay is loaded once, or a supplied `overlay_sf` object is used.
9. **Marine-region assignment.** Occurrences are assigned by `sf::st_within()`, with an `sf::st_intersects()` fallback for coastal or boundary records.
10. **Cleaning and thinning.** Export groups are optionally cleaned and spatially thinned.
11. **Export and auditing.** Results are written to CSV and/or returned as a combined in-memory table, with GBIF retrieval outcomes retained separately from downstream spatial-processing summaries.

Value

If `return_all_results = TRUE` and `store_in_memory = TRUE`, returns a tibble containing the combined processed marine occurrence records from all retained species x marine-region groups. The returned table includes occurrence columns imported from GBIF, restored `decimalLongitude` and `decimalLatitude` columns, marine-region assignment fields such as `region_id`, `region_name`, `region_type`, `workflow`, `marine_region_id`, `marine_region_name`, `marine_region_source`, `marine_regions_layer` and `geoname` where available, plus optional taxonomy, GRIIS and native-origin audit columns when those evidence gates are enabled. If `return_all_results = FALSE` or `store_in_memory = FALSE`, the function writes per-group CSV files and summary/audit outputs to `output_dir` and returns `invisible(NULL)`. In all modes, the primary side effects are GBIF download/import operations, marine overlay assignment, optional cleaning/thinning and writing workflow outputs.

GBIF download sequencing and retrieval auditing

`batch_size` controls only the outer grouping of species. It does not define the number of simultaneous asynchronous GBIF downloads.

Within every outer batch, the pipeline deliberately uses:
`submit species -> wait/poll -> import -> reconcile -> next species`.

This ordering prevents a nominal batch containing several species from submitting all of those GBIF jobs before any one has completed.

Each retained species receives one row in `gbif_retrieval_audit.csv`. The audit contains `species`, `request_id`, `request_type`, `gbif_key`, `gbif_status`, `retrieval_status`, `n_records`, and `fail_reason`. For the marine workflow, `request_type` is `"species"` and the request identifier represents the species-level GBIF retrieval.

Successfully completed downloads are recorded as `success` or `success_zero`. Unresolved or failed states are preserved explicitly rather than being interpreted as zero occurrences.

Marine GRIIS and native-origin joins are species-level by default because marine recipient units are usually polygons rather than ISO country records. These columns should therefore be interpreted as species-level evidence unless a separate recipient-specific origin table is supplied elsewhere in the workflow.

Supported overlay names are provided by `bf_marine_overlay_sources()` and loaded by `bf_load_marine_regions_overl`. Typical retained overlays include EEZs, Marine Ecoregions of the World, Large Marine Ecosystems, IHO sea areas, high seas, territorial seas, internal waters, archipelagic waters, Longhurst provinces and UNESCO marine World Heritage sites, depending on the installed Marine Regions/mregions2 data products.

The `deps` argument is for testing and advanced use. It can inject mock or alternative implementations of summary, download, import, export or thinning functions without changing the production pipeline.

Output files

When `store_in_memory = FALSE` or when export helpers are active, the pipeline writes one CSV per retained species x marine-region group. The output directory can also contain `gbif_summary.csv`, `gbif_retrieval_audit.csv`, `gbif_unresolved_requests.csv`, and optional audit files for the taxonomy, GRIIS, native-origin and native-web evidence gates. The retrieval audit records one explicit GBIF outcome per retained species. Successful, empty, failed and unresolved species therefore remain distinguishable independently of the number of marine-region output groups produced downstream. Unresolved requests retain their GBIF keys where available for targeted review or recovery.

Data access, licensing and attribution

This function can submit live GBIF downloads and can use third-party marine overlay and native-evidence resources. biofetchR does not redistribute those provider datasets from this script; it caches or exports user-side results created during the workflow. Users should retain GBIF download keys/DOIs and cite GBIF, Marine Regions or other overlay providers, GRIIS/native-evidence sources and any supplied spatial layers according to the licence and citation requirements of the exact data products used.

Failure handling

Download submission, polling, import, sf-conversion, overlay matching and export failures are recorded explicitly where possible.

GBIF retrieval outcomes are maintained independently in `gbif_retrieval_audit.csv`. Unresolved or failed statuses copied to `gbif_unresolved_requests.csv` include `taxon_key_failed`, `submit_failed`, `submit_no_key`, `submit_timeout`, `invalid_key`, `pending_timeout`, `download_failed`, `killed`, `cancelled`, `file_erased`, `import_failed`, `split_failed`, and `internal_unreconciled`.

`pending_timeout` means that the submitted GBIF download did not reach a terminal state within the configured polling window; its key is retained and the request is not interpreted as a zero-occurrence result.

A final species-level completeness gate checks that every retained marine species received a retrieval outcome. Any missing species is recorded as `internal_unreconciled` rather than silently disappearing.

Fatal configuration errors, such as missing required packages or unsupported overlay names, still stop early before avoidable GBIF requests are submitted.

See Also

`download_gbif_batch()`, `wait_and_import_gbif_safe()`, `thin_spatial_points()`, `bf_load_marine_regions_overl`

Other marine processing pipelines: `process_gbif_eez_pipeline()`

Examples

```
gbif_env <- c("GBIF_USER", "GBIF_PWD", "GBIF_EMAIL")

if (interactive() && all(nzchar(Sys.getenv(gbif_env)))) {
  marine_input <- data.frame(
    species = c("Ficopomatus enigmaticus", "Carcinus maenas")
  )

  process_gbif_marine_pipeline(
    df = marine_input,
    output_dir = file.path(tempdir(), "marine_gbif_outputs"),
    user = Sys.getenv("GBIF_USER"),
    pwd = Sys.getenv("GBIF_PWD"),
    email = Sys.getenv("GBIF_EMAIL"),
    overlay = "meow",
    prepare_taxonomy = TRUE,
    apply_cleaning = TRUE,
    apply_thinning = FALSE
  )
}
```

---

process_gbif_terrestrial_freshwater_pipeline

Process terrestrial and freshwater GBIF occurrences

Description

`process_gbif_terrestrial_freshwater_pipeline()` is the main high-level terrestrial/freshwater occurrence workflow in biofetchR. It applies optional taxonomy and origin-evidence gates before GBIF submission, pre-checks retained species-country requests for georeferenced records, submits one combined multi-country download per species, imports that archive once, and reconstructs country-specific occurrence objects before the existing context, overlay, cleaning, thinning and export stages.

Retrieval provenance is retained separately from downstream spatial-output summaries so every original request can be classified explicitly as successful, empty, failed, or unresolved.

Usage

```
process_gbif_terrestrial_freshwater_pipeline(
  df,
  output_dir,
  user,
  pwd,
  email,
  region_source = "gadm",
  raster_context = character(0),
  overlay_mode = c("single", "dual_separate", "dual_intersection"),
  gadm_unit = 1,
  cache_dir_gadm = NULL,
  teow_cache_dir = NULL,
  teow_method = c("auto", "mapme", "direct"),
  teow_url = NULL,
  teow_force_refresh = FALSE,
  teow_clip_to_countries = TRUE,
  feow_cache_dir = NULL,
  feow_method = c("auto", "mapme", "direct"),
  feow_url = NULL,
  feow_force_refresh = FALSE,
  feow_clip_to_countries = TRUE,
  lakes_cache_dir = NULL,
  lakes_force_refresh = FALSE,
  lakes_only = TRUE,
  lakes_min_area_km2 = 1,
  rivers_cache_dir = NULL,
  rivers_cache = c("disk", "memory"),
  rivers_cache_format = c("rds", "gpkg", "both"),
  rivers_force_refresh = FALSE,
  rivers_regions = NULL,
  rivers_min_strahler = NULL,
  rivers_min_discharge_cms = NULL,
  rivers_max_snap_m = 1000,
  basins_cache_dir = NULL,
  basins_level = 12,
  basins_with_lakes = FALSE,
  basins_clip_to_countries = TRUE,
  gmba_cache_dir = NULL,
  gmba_layer = c("all", "basic"),
  gmba_extent = c("standard", "broad"),
  gmba_force_refresh = FALSE,
  gmba_clip_to_countries = TRUE,
  ne_cache_dir = NULL,
  ne_force_refresh = FALSE,
  ne_admin1_scale = "10m",
  resolve_cache_dir = NULL,
  resolve_force_refresh = FALSE,
```

```
  resolve_clip_to_countries = TRUE,
  wdpa_cache_dir = NULL,
  wdpa_force_refresh = FALSE,
  wdpa_exclude_marine = TRUE,
  wdpa_require_opt_in = TRUE,
  wdpa_opt_in = FALSE,
  ramsar_cache_dir = NULL,
  ramsar_force_refresh = FALSE,
  ramsar_require_opt_in = TRUE,
  ramsar_opt_in = FALSE,
  gdw_cache_dir = NULL,
  gdw_force_refresh = FALSE,
  gdw_barriers_source_path = NULL,
  gdw_barriers_source_url = NULL,
  gdw_barrier_max_snap_m = 1000,
  biosphere_cache_dir = NULL,
  biosphere_force_refresh = FALSE,
  biosphere_source_path = NULL,
  biosphere_source_url = NULL,
  biosphere_max_snap_m = 10000,
  gloric_cache_dir = NULL,
  gloric_force_refresh = FALSE,
  gloric_source_path = NULL,
  gloric_source_url = NULL,
  gloric_max_snap_m = 1000,
  hydrowaste_cache_dir = NULL,
  hydrowaste_force_refresh = FALSE,
  hydrowaste_source_path = NULL,
  hydrowaste_source_url = NULL,
  hydrowaste_max_snap_m = 1000,
  gdw_reservoirs_cache_dir = NULL,
  gdw_reservoirs_force_refresh = FALSE,
  gdw_reservoirs_source_path = NULL,
  gdw_reservoirs_source_url = NULL,
  global_mining_cache_dir = NULL,
  global_mining_force_refresh = FALSE,
  global_mining_source_path = NULL,
  global_mining_source_url = NULL,
  strict_overlay_loading = FALSE,
  strict_raster_context = FALSE,
  raster_cache_dir = NULL,
  worldcover_source = NULL,
  worldcover_year = 2021,
  worldcover_buffer_m = 0,
  soilgrids_sources = NULL,
  soilgrids_vars = NULL,
  soilgrids_buffer_m = 0,
  human_footprint_source = NULL,
```

```
  human_footprint_buffer_m = 0,
  batch_size = 5,
  dist_km = 5,
  apply_thinning = FALSE,
  apply_cleaning = TRUE,
  return_all_results = TRUE,
  export_summary = TRUE,
  store_in_memory = TRUE,
  use_planar = TRUE,
  use_overlays = TRUE,
  prepare_taxonomy = FALSE,
  manual_taxonomy_fixes = NULL,
  taxonomy_name_col = "species",
  export_taxonomy_audit = TRUE,
  quiet = FALSE,
  deps = NULL,
  use_griis = FALSE,
  filter_griis_invasive = FALSE,
  griis = NULL,
  griis_cache_dir = NULL,
  griis_force_refresh = FALSE,
  griis_require_country = NULL,
  use_native_range = FALSE,
  native_ranges = NULL,
  use_native_web = FALSE,
  native_web_sources = c("sinas", "gbif", "worms"),
  native_web_cache_dir = NULL,
  native_web_force_refresh = FALSE,
  native_web_sleep_sec = 0.25,
  native_web_sinas_main_path = NULL,
  native_web_sinas_alllocations_path = NULL,
  native_web_sinas_fulltaxa_path = NULL,
  export_native_web_audit = TRUE,
  native_species_col = "species",
  native_require_country = NULL,
 native_filter_mode = c("audit_only", "non_native_only", "non_native_or_unknown",
    "native_only"),
  native_keep_unknown = TRUE,
  reconcile_origin_evidence = TRUE,
  export_origin_audit = TRUE
)
```

Arguments

df	Data frame containing at least species and iso2c.
output_dir	Directory where occurrence CSVs, gbif_summary.csv, gbif_retrieval_audit.csv, gbif_unresolved_requests.csv, and optional taxonomy/origin audit files are written.

`user` GBIF username.

`pwd` GBIF password.

`email` GBIF account email address.

`region_source` Character vector of terrestrial/freshwater vector overlay sources. See **Supported vector overlays** for accepted values.

`raster_context` Optional character vector of raster/context enrichments. Accepted values are `"worldcover"`, `"soilgrids"`, and `"human_footprint"`. These enrich occurrence points with attributes and do not create grouped outputs by themselves.

`overlay_mode` Overlay combination mode. `"single"` processes each requested overlay independently. `"dual_separate"` keeps requested overlays separate. `"dual_intersection"` exports groups based on combined overlay labels when multiple overlays are requested.

`gadm_unit` Integer GADM administrative level used when `region_source` includes `"gadm"`. Supported values are `0`, `1`, and `2`.

`cache_dir_gadm` Directory for cached GADM objects. If `NULL`, a subdirectory under `output_dir/_biofetchR_cache/` is used.

`teow_cache_dir`, `teow_method`, `teow_url`
TEOW loading and source settings.

`teow_force_refresh`, `teow_clip_to_countries`
TEOW refresh and clipping settings. If `teow_cache_dir = NULL`, a subdirectory under `output_dir/_biofetchR_cache/` is used.

`feow_cache_dir`, `feow_method`, `feow_url`
FEOW loading and source settings.

`feow_force_refresh`, `feow_clip_to_countries`
FEOW refresh and clipping settings. If `feow_cache_dir = NULL`, a subdirectory under `output_dir/_biofetchR_cache/` is used.

`lakes_cache_dir`, `lakes_force_refresh`
HydroLAKES loading and caching settings.

`lakes_only`, `lakes_min_area_km2`
HydroLAKES filtering settings. If `lakes_cache_dir = NULL`, a subdirectory under `output_dir/_biofetchR_cache/` is used.

`rivers_cache_dir`, `rivers_cache`, `rivers_cache_format`
HydroRIVERS caching settings.

`rivers_force_refresh`, `rivers_regions`
HydroRIVERS refresh and regional loading settings.

`rivers_min_strahler`, `rivers_min_discharge_cms`
HydroRIVERS filtering settings.

`rivers_max_snap_m`
Maximum snapping distance in metres for assigning points to nearest HydroRIVERS reaches. If `rivers_cache_dir = NULL`, a subdirectory under `output_dir/_biofetchR_cache/` is used.

`basins_cache_dir`, `basins_level`
HydroBASINS loading settings.

basins_with_lakes, basins_clip_to_countries
: HydroBASINS filtering and clipping settings. If basins_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

gmba_cache_dir, gmba_layer, gmba_extent
: GMBA loading settings.

gmba_force_refresh, gmba_clip_to_countries
: GMBA refresh and clipping settings. If gmba_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

ne_cache_dir, ne_force_refresh
: Natural Earth loading and caching settings.

ne_admin1_scale
: Natural Earth administrative-unit scale. If ne_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

resolve_cache_dir, resolve_force_refresh
: RESOLVE 2017 loading and caching settings.

resolve_clip_to_countries
: RESOLVE 2017 clipping setting. If resolve_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

wdpa_cache_dir, wdpa_force_refresh
: WDPA loading and caching settings.

wdpa_exclude_marine
: WDPA filtering setting controlling whether marine protected areas are excluded.

wdpa_require_opt_in, wdpa_opt_in
: WDPA opt-in controls. WDPA loading is opt-in by default because this source can be large and may require user awareness of provider terms. If wdpa_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

ramsar_cache_dir, ramsar_force_refresh
: Ramsar loading and caching settings.

ramsar_require_opt_in, ramsar_opt_in
: Ramsar opt-in controls. Ramsar loading is opt-in by default. If ramsar_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

gdw_cache_dir, gdw_force_refresh
: Global Dam Watch barrier loading and caching settings.

gdw_barriers_source_path, gdw_barriers_source_url
: Optional local path or URL for Global Dam Watch barrier data.

gdw_barrier_max_snap_m
: Maximum snapping distance in metres for Global Dam Watch barrier assignment. If gdw_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

biosphere_cache_dir, biosphere_force_refresh
: UNESCO biosphere-reserve loading and caching settings.

biosphere_source_path, biosphere_source_url
: Optional local path or URL for biosphere-reserve data.

biosphere_max_snap_m
: Maximum snapping distance in metres for biosphere-reserve assignment. If biosphere_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

gloric_cache_dir, gloric_force_refresh
: GLORIC loading and caching settings.

gloric_source_path, gloric_source_url
: Optional local path or URL for GLORIC data.

gloric_max_snap_m
: Maximum snapping distance in metres for GLORIC assignment. If gloric_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

hydrowaste_cache_dir, hydrowaste_force_refresh
: HydroWASTE loading and caching settings.

hydrowaste_source_path, hydrowaste_source_url
: Optional local path or URL for HydroWASTE data.

hydrowaste_max_snap_m
: Maximum snapping distance in metres for HydroWASTE assignment. If hydrowaste_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

gdw_reservoirs_cache_dir, gdw_reservoirs_force_refresh
: Global Dam Watch reservoir loading and caching settings.

gdw_reservoirs_source_path, gdw_reservoirs_source_url
: Optional local path or URL for Global Dam Watch reservoir data. If gdw_reservoirs_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

global_mining_cache_dir, global_mining_force_refresh
: Global mining-area loading and caching settings.

global_mining_source_path, global_mining_source_url
: Optional local path or URL for global mining-area data. If global_mining_cache_dir = NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

strict_overlay_loading
: Logical. If TRUE, a requested overlay-loading failure stops the pipeline. If FALSE, the failure is recorded in gbif_summary.csv and processing continues where possible.

strict_raster_context
: Logical. If TRUE, a requested raster/context enrichment failure stops the pipeline. If FALSE, the failure is recorded and processing continues with unenriched points.

raster_cache_dir
: Directory for raster/context caches. If NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

worldcover_source
: Path, URL or supported raster object for WorldCover extraction.

worldcover_year
: Year label used for WorldCover output metadata.

worldcover_buffer_m
: Buffer radius in metres for WorldCover extraction. 0 uses point extraction.

soilgrids_sources
: Named vector/list of paths, URLs or supported raster objects keyed by SoilGrids variable name.

soilgrids_vars
: Character vector of SoilGrids variable names to extract; must match names in soilgrids_sources.

`soilgrids_buffer_m`
: Buffer radius in metres for SoilGrids extraction. `0` uses point extraction.

`human_footprint_source`
: Path, URL or supported raster object for Human Footprint extraction.

`human_footprint_buffer_m`
: Buffer radius in metres for Human Footprint extraction. `0` uses point extraction.

`batch_size`
: Number of species grouped into each outer processing batch. Species are still processed sequentially within those batches. With the built-in terrestrial/freshwater backend, all eligible countries for one species are combined into one GBIF download; `batch_size` therefore does not represent the number of concurrent country-level GBIF jobs.

`dist_km`
: Minimum distance in kilometres used when `apply_thinning = TRUE`.

`apply_thinning`
: Logical. If `TRUE`, apply distance-based thinning after coordinate cleaning.

`apply_cleaning`
: Logical. If `TRUE`, apply coordinate-quality cleaning before export.

`return_all_results`
: Logical. If `TRUE` and `store_in_memory = TRUE`, return a combined tibble of all grouped records.

`export_summary`
: Logical. If `TRUE`, write `gbif_summary.csv`, `gbif_retrieval_audit.csv`, and `gbif_unresolved_requests.csv`.

`store_in_memory`
: Logical. If `TRUE`, retain grouped output rows in memory for return.

`use_planar`
: Logical. If `TRUE`, disable spherical geometry for selected spatial operations during this run.

`use_overlays`
: Logical. If `FALSE`, skip vector overlay assignment and export records grouped at the country/raw-occurrence level.

`prepare_taxonomy`
: Logical. If `TRUE`, run the taxonomy gate before GBIF download submission.

`manual_taxonomy_fixes`
: Optional named character vector or data frame of manual taxonomy fixes applied before automated cleaning.

`taxonomy_name_col`
: Name of the column containing submitted taxon names.

`export_taxonomy_audit`
: Logical. If `TRUE`, write taxonomy audit files to `output_dir`.

`quiet`
: Logical. If `TRUE`, reduce console messages.

`deps`
: Optional named list of dependency overrides for tests.

`use_griis`
: Logical. If `TRUE`, attach GRIIS evidence before GBIF submission.

`filter_griis_invasive`
: Logical. If `TRUE`, retain only rows flagged as invasive by GRIIS in the recipient country. Requires `use_griis = TRUE`.

`griis`
: Optional pre-read or standardised GRIIS table.

`griis_cache_dir`
: Cache directory used when GRIIS is loaded internally. If `NULL`, a subdirectory under `output_dir/_biofetchR_cache/` is used.

griis_force_refresh
: Logical. If TRUE, force GRIIS refresh when loaded internally.

griis_require_country
: Optional logical controlling whether GRIIS joins require recipient-country evidence. Defaults to TRUE in this pipeline.

use_native_range
: Logical. If TRUE, attach native-range evidence before GBIF submission.

native_ranges
: Native-range evidence table used by bf_attach_native_status(). Leave NULL when use_native_web = TRUE so the pipeline can build species-level native-origin evidence from web/API sources after taxonomy and GRIIS filtering.

use_native_web
: Logical. If TRUE, fetch native-origin evidence from package web/API helpers before applying the native-range gate. Requires use_native_range = TRUE and native_ranges = NULL.

native_web_sources
: Character vector of native web sources passed to bf_fetch_native_ranges_web(), commonly c("sinas", "gbif", "worms").

native_web_cache_dir
: Cache directory for native web/API responses. If NULL, a subdirectory under output_dir/_biofetchR_cache/ is used.

native_web_force_refresh
: Logical. If TRUE, refresh native-web cache entries where supported.

native_web_sleep_sec
: Numeric delay in seconds between native-web requests.

native_web_sinas_main_path
: Optional local path to SInAS_3.1.1.csv when "sinas" is included in native_web_sources.

native_web_sinas_alllocations_path
: Optional local path to AllLocations.xlsx, .csv or .tsv.

native_web_sinas_fulltaxa_path
: Optional local path to SInAS_3.1.1_FullTaxaList.csv.

export_native_web_audit
: Logical. If TRUE, write native-web long, species, unmapped and summary audit files to output_dir.

native_species_col
: Species-name column in native_ranges.

native_require_country
: Optional logical controlling whether native-range joins require recipient-country evidence. Defaults to TRUE in this pipeline.

native_filter_mode
: Native-range filter mode. Accepted values are "audit_only", "non_native_only", "non_native_or_unknown", and "native_only".

native_keep_unknown
: Logical. Used with native_filter_mode = "non_native_only"; if TRUE, rows with unknown origin are retained.

reconcile_origin_evidence
: Logical. If TRUE, reconcile attached GRIIS and native-range evidence into a combined origin-evidence status where the required helper is available.

`export_origin_audit`
: Logical. If `TRUE`, write origin-evidence audit files to `output_dir`.

Details

Download, import, clean, assign and export GBIF occurrence records for terrestrial and freshwater workflows. Input is supplied as a species x ISO2 country table. Species-country combinations remain the logical retrieval, auditing and output units, but eligible countries for one species are combined into a single asynchronous GBIF occurrence download whenever the built-in multi-country backend is used.

The pipeline runs in the following order:

1. **Input validation and standardisation.** The input table must contain a species column and an ISO2 country column. Species names are coerced to character and ISO2 codes are uppercased.
2. **Optional taxonomy gate.** When `prepare_taxonomy = TRUE`, names are cleaned, manual fixes can be applied, invalid/non-species entries can be rejected, and accepted names are used downstream.
3. **Optional origin-evidence gates.** GRIIS and native-range evidence can be attached and used to retain or reject species-country rows before GBIF is queried.
4. **Country-level GBIF availability pre-check.** Each retained species-country combination is checked for georeferenced GBIF records. A confirmed zero is recorded explicitly as `precheck_zero` and is not included in the download.
5. **Multi-country GBIF submission.** For each species, all countries retained after the pre-check are submitted together to the built-in backend as one asynchronous GBIF download using a country-IN predicate.
6. **GBIF polling, import and country reconstruction.** The shared download is polled and imported once. Its records are split locally by GBIF `countryCode`, recreating one result object for every requested country.
7. **Retrieval reconciliation.** Every retained species-country request is reconciled against the submission/import audit. Successful empty subsets, failures, timeouts and internally missing outcomes remain explicit.
8. **Optional raster/context enrichment.** Requested raster/context variables are extracted to occurrence points as additional attributes.
9. **Vector overlay assignment.** Points are assigned to requested terrestrial/freshwater spatial units using polygon containment or nearest-feature snapping, depending on overlay type.
10. **Coordinate cleaning and thinning.** Coordinate-quality cleaning is applied when `apply_cleaning = TRUE`; distance-based thinning is applied when `apply_thinning = TRUE` and `dist_km > 0`.
11. **Export and auditing.** Grouped occurrence CSVs, processing summaries and GBIF retrieval-audit files are written to `output_dir`. If `store_in_memory = TRUE`, grouped records can also be returned as a combined tibble.

Value

If `return_all_results = TRUE` and `store_in_memory = TRUE`, returns a tibble containing the combined processed terrestrial/freshwater occurrence records from all retained species x spatial-recipient

groups. The returned table has geometry dropped and includes restored `decimalLongitude` and `decimalLatitude` columns, GBIF occurrence fields, grouping fields such as species, recipient region identifiers and region type, and any overlay, raster-context, taxonomy, GRIIS or native-origin audit columns created by the selected workflow options. If no rows remain after taxonomy or origin-evidence filtering, the returned tibble contains the stable empty coordinate columns `decimalLongitude` and `decimalLatitude`. If `return_all_results = FALSE` or `store_in_memory = FALSE`, the function writes grouped occurrence CSVs, `gbif_summary.csv` and any requested audit files to `output_dir` and returns `invisible(NULL)`. In all modes, the main side effects are GBIF download/import operations, optional raster/context enrichment, vector overlay assignment, coordinate cleaning, optional spatial thinning and writing workflow outputs.

GBIF multi-country retrieval architecture

The input and audit unit remains a species x ISO2-country combination. The asynchronous GBIF download unit is different: when the built-in `download_gbif_batch_gadm()` backend is used, all retained countries for one species are combined in a single country-IN request.

Consequently, several countries for the same species normally carry the SAME GBIF download key. This is intentional. The completed archive is imported only once and `wait_and_import_gbif()` splits it back to country-specific objects using GBIF `countryCode`.

This design reduces unnecessary asynchronous download jobs while preserving country-level outputs and provenance. A successful country split containing no records is retained as `success_zero`; it is not silently omitted.

Input table

`df` must contain at least:

- `species`: submitted taxon name or common-name/manual-fix input, depending on the taxonomy settings.
- `iso2c`: recipient ISO2 country code used for terrestrial/freshwater GBIF country predicates.

Additional columns are retained where possible and can be carried into summaries and audit files as metadata.

Supported vector overlays

Use `region_source` to request one or more overlays. Supported values are `"gadm"`, `"ne_admin1"`, `"ne_urban"`, `"resolve2017"`, `"teow"`, `"feow"`, `"basins"`, `"lakes"`, `"rivers"`, `"gmba"`, `"wdpa"`, `"ramsar"`, `"gdw_barriers"`, `"biosphere_reserve"`, `"gloric"`, `"hydrowaste"`, `"gdw_reservoirs"`, and `"global_mining"`.

Polygon overlays assign points by spatial intersection. Line or point-like resources such as rivers, barriers, HydroWASTE, biosphere reserves and GLORIC use nearest-feature assignment with overlay-specific maximum snapping distances.

Origin-evidence gates

GRIIS and native-range evidence are optional and are evaluated before GBIF download submission. For terrestrial/freshwater workflows, country-aware matching is the default: GRIIS and native-range evidence are interpreted for each species in the recipient country when the relevant data are available. Audit files can be written by setting `export_origin_audit = TRUE`.

Cleaning and thinning

Coordinate cleaning and thinning are delegated to `thin_spatial_points()`. Cleaning removes invalid, impossible, missing and zero-zero coordinates and can filter records with high coordinate uncertainty. Thinning is distance-based and prioritises higher-quality records before removing nearby clustered records.

Output files

The pipeline can write:

- grouped occurrence CSVs for each species-region combination;
- `gbif_summary.csv`, recording processing status, row counts, download keys and failure stages;
- `gbif_retrieval_audit.csv`, recording one explicit GBIF retrieval outcome for each taxonomy-approved species x country request;
- `gbif_unresolved_requests.csv`, containing only unresolved or failed GBIF retrieval requests for targeted review/recovery;
- `taxonomy_audit.csv`, `taxonomy_rejected.csv` and `taxonomy_summary.csv` when taxonomy auditing is enabled;
- `origin_evidence_audit.csv`, `origin_evidence_rejected.csv` and `origin_evidence_summary.csv` when origin-evidence auditing is enabled.

Retrieval audit and completeness

`gbif_retrieval_audit.csv` is deliberately separate from `gbif_summary.csv`. The processing summary can contain multiple rows per original request after spatial overlay assignment, whereas the retrieval audit contains one row per retained GBIF request unit.

The retrieval audit contains:

- `species`;
- `request_id` (ISO2 country);
- `request_type` (`"country"`);
- `gbif_key`;
- `gbif_status`;
- `retrieval_status`;
- `n_records`;
- `fail_reason`.

Resolved retrieval outcomes include `success`, `success_zero`, and `precheck_zero`. Unresolved or failed states written to `gbif_unresolved_requests.csv` include `taxon_key_failed`, `submit_failed`, `submit_no_key`, `submit_timeout`, `invalid_key`, `pending_timeout`, `download_failed`, `killed`, `cancelled`, `file_erased`, `import_failed`, `split_failed`, and `internal_unreconciled`.

The final completeness gate compares the retained expected species-country requests with the retrieval ledger. Any request lacking an explicit outcome is added as `internal_unreconciled` rather than being silently omitted.

Data access, licences and attribution

This function can submit live GBIF occurrence downloads and can call helper functions that load third-party vector and raster resources. biofetchR does not redistribute those provider datasets from this pipeline, but users are responsible for complying with the licence, citation and attribution terms of every dataset requested in a run. Retain GBIF download keys and cite the corresponding GBIF download DOI where GBIF-mediated records are used.

Licence-sensitive or attribution-sensitive resources may include, depending on the arguments used, GBIF, GADM, Natural Earth, HydroSHEDS/HydroBASINS, HydroLAKES, HydroRIVERS, TEOW, FEOW, RESOLVE ecoregions, WDPA, Ramsar, GMBA, Global Dam Watch, HydroWASTE, global mining layers, ESA WorldCover, SoilGrids, Human Footprint and user-supplied spatial or raster layers. Always cite the exact product, version, year and provider used in the workflow.

Failure handling

Overlay and raster-context failures are handled according to `strict_overlay_loading` and `strict_raster_context`. In fail-soft mode these failures are recorded in `gbif_summary.csv` and processing continues where possible.

GBIF retrieval failures use a separate request-level ledger. Submission failures, invalid keys, download terminal states, polling timeouts, import failures, split failures and completeness failures are written to `gbif_retrieval_audit.csv`. Unresolved states are also copied to `gbif_unresolved_requests.csv`.

A polling timeout is recorded as `pending_timeout` with the GBIF key retained. It is not treated as evidence of zero occurrence records. Likewise, a request that cannot be reconciled to either an imported result or an explicit failure is recorded as `internal_unreconciled`.

Strict overlay/context modes can still stop the pipeline immediately for configuration or resource failures.

Testing and dependency injection

The `deps` argument allows tests to inject mock download, import, export, summary, overlay and thinning helpers. Normal users should usually leave `deps = NULL`; package tests can use it to exercise the pipeline without submitting live GBIF downloads or requiring large remote overlay resources.

See Also

`thin_spatial_points()`, `download_gbif_batch_gadm()`, `bf_prepare_taxa_for_gbif()`, `bf_attach_griis_status()`
`bf_attach_native_status()`

`resolve_species_names` *Resolve and standardise species names*

Description

Backward-compatible wrapper around `bf_resolve_gbif_taxonomy_batch()`.

Usage

```
resolve_species_names(
  names,
  sources = c("GBIF", "ITIS"),
  use_taxize = TRUE,
  verbose = TRUE
)
```

Arguments

names	Character vector of species names.
sources	Retained for compatibility. Currently not used by the GBIF-first resolver.
use_taxize	Retained for compatibility.
verbose	Print progress messages.

Value

A tibble with one row per input name. The output includes backward-compatible columns original_name, resolved_name, match_type, source and score, followed by the full GBIF-first taxonomy-resolution columns returned by bf_resolve_gbif_taxonomy_batch(). The table provides a compatibility interface for older workflows while retaining the current biofetchR taxonomy audit fields.

---

thin_spatial_points *Clean and spatially thin GBIF-style occurrence points*

Description

thin_spatial_points() performs conservative coordinate screening and optional spatial thinning for GBIF-style occurrence records. It accepts either an sf point object or a plain data frame/tibble containing decimalLongitude and decimalLatitude columns.

Usage

```
thin_spatial_points(
  sf_obj,
  dist_km = 5,
  priority_col = "individualCount",
  return_all = FALSE,
  plot = FALSE,
  max_density = NULL,
  max_uncertainty_m = 10000,
  warn_uncertainty_m = 5000,
  filter_uncertain = TRUE,
  quiet = FALSE
)
```

Arguments

sf_obj	An sf point object, or a data frame/tibble containing decimalLongitude and decimalLatitude columns.
dist_km	Numeric. Minimum distance, in kilometres, between retained points during distance-based thinning. Use dist_km = 0 for cleaning-only behaviour.
priority_col	Character. Name of a column used to prioritise retained records during thinning. Higher values are retained first. If the column is absent, all points are treated as equal priority. Defaults to "individualCount".
return_all	Logical. If TRUE, return all post-cleaning records and add .thinned to indicate whether each record was retained. If FALSE, return only retained records. Defaults to FALSE.
plot	Logical. If TRUE and **ggplot2** is installed, print a diagnostic retained-versus-removed point plot. Defaults to FALSE.
max_density	Optional numeric. If supplied, apply density-based thinning by retaining at most this many points per km^2 within the input bounding box. If NULL, distance-based thinning is used. Defaults to NULL.
max_uncertainty_m	Numeric. Maximum accepted coordinate uncertainty, in metres, when filter_uncertain = TRUE. Defaults to 10000.
warn_uncertainty_m	Numeric. Uncertainty threshold, in metres, above which retained records are flagged in .high_uncertainty. Defaults to 5000.
filter_uncertain	Logical. If TRUE, remove records with coordinateUncertaintyInMeters > max_uncertainty_m. If FALSE, retain those records but still coerce uncertainty to numeric and use it as a small priority penalty during thinning. Defaults to TRUE.
quiet	Logical. If TRUE, suppress console messages. Defaults to FALSE.

Details

This function is used internally by the main biofetchR pipelines, but it is also exported so users can apply the same cleaning and thinning logic to occurrence tables before, after or outside a full pipeline run.

The function applies its checks in a fixed order:

1. **Input standardisation**. A plain data frame is screened for usable longitude/latitude values and converted to an sf object in EPSG:4326. Existing sf inputs are retained as sf objects.
2. **Coordinate screening**. Records with missing, non-finite, impossible coordinates, or exact 0,0 coordinates, are removed.
3. **Coordinate-uncertainty handling**. If coordinateUncertaintyInMeters is present, it is coerced to numeric. When filter_uncertain = TRUE, records above max_uncertainty_m are removed. When filter_uncertain = FALSE, those records are retained but uncertainty still acts as a small priority penalty during thinning.

4. **Priority ordering**. Records are sorted before thinning so that higher values in `priority_col` are preferentially retained. By default, `individualCount` is used when present.
5. **Spatial thinning**. If `max_density` is supplied, density-based thinning is applied. Otherwise, if `dist_km > 0`, greedy distance-based thinning is applied using great-circle distances from `geosphere::distHaversine()`.

Set `dist_km = 0` and leave `max_density = NULL` to use the function as a cleaning-only helper. In that mode no distance thinning is applied, and all cleaned records are returned.

Value

An `sf` point object containing occurrence records that passed the coordinate-cleaning rules and, when thinning is requested, the spatial thinning rules. The returned object retains `decimalLongitude` and `decimalLatitude` columns for downstream CSV export, diagnostics and spatial joins. If `return_all = TRUE`, the output includes a logical `.thinned` column, where `TRUE` marks records retained by the thinning step and `FALSE` marks records removed by thinning. If `coordinateUncertaintyInMeters` is present, the output may also include a logical `.high_uncertainty` column indicating records with uncertainty above `warn_uncertainty_m`. When no valid records remain, an empty `sf` object is returned with the same output conventions.

Input requirements

`sf_obj` may be either:

- an `sf` point object; or
- a data frame/tibble with `decimalLongitude` and `decimalLatitude` columns.

Non-`sf` inputs are converted to `sf` with EPSG:4326 and with longitude and latitude columns retained (`remove = FALSE`). Existing `sf` inputs are assumed to represent point geometries with longitude/latitude-like coordinates.

Cleaning rules

The coordinate-cleaning step removes records with:

- missing longitude or latitude;
- non-finite longitude or latitude;
- longitude outside `[-180, 180]`;
- latitude outside `[-90, 90]`; or
- exact `0,0` coordinates.

If `coordinateUncertaintyInMeters` is present and `filter_uncertain = TRUE`, records with uncertainty greater than `max_uncertainty_m` are also removed.

Thinning behaviour

Distance-based thinning is greedy and order-dependent. The function therefore sorts records before thinning so that higher-priority records are considered first. If `coordinateUncertaintyInMeters` is available, higher uncertainty reduces the thinning priority slightly, even when high-uncertainty records are not filtered out.

Density-based thinning is activated by `max_density`. In that mode, the function estimates the input bounding-box area and retains approximately `max_density` records per km^2, after priority sorting.

Output columns

The returned object is always an `sf` object. The function ensures that `decimalLongitude` and `decimalLatitude` are available in the output. If `return_all = TRUE`, a logical `.thinned` column indicates which rows were retained by the thinning step. If `coordinateUncertaintyInMeters` is present, a logical `.high_uncertainty` column flags retained records above `warn_uncertainty_m`.

Workflow note

This function is intentionally conservative and local-only. It does not perform taxonomic cleaning, native-range filtering, GRIIS filtering, GBIF download submission or environmental overlay extraction. Those steps belong to higher-level biofetchR pipeline functions.

See Also

`sf::st_as_sf()`, `sf::st_coordinates()`, `geosphere::distHaversine()`

Examples

```
pts <- data.frame(
  species = "Example species",
  decimalLongitude = c(-6.26, -6.2601, 0, 181),
  decimalLatitude = c(53.35, 53.3501, 0, 53),
  coordinateUncertaintyInMeters = c(20, 30, 10, 10),
  individualCount = c(5, 1, 1, 1)
)

cleaned <- thin_spatial_points(
  pts,
  dist_km = 0,
  filter_uncertain = TRUE,
  quiet = TRUE
)

thinned <- thin_spatial_points(
  pts,
  dist_km = 5,
  priority_col = "individualCount",
  filter_uncertain = TRUE,
  quiet = TRUE
)
```

`wait_and_import_gbif` *Wait for GBIF occurrence downloads and import completed records*

Description

Poll submitted GBIF occurrence-download keys, import successfully completed SIMPLE_CSV archives, convert coordinate-bearing records to WGS84 `sf` points, and preserve one explicit retrieval outcome for every original request label.

Usage

```
wait_and_import_gbif(keys, wait_time = 30, max_tries = 60)
```

Arguments

`keys`	Named list or named character vector of GBIF occurrence-download keys. Names are logical request labels. Several names may intentionally map to the same key.
`wait_time`	Numeric. Seconds between GBIF metadata polling attempts.
`max_tries`	Integer. Maximum number of metadata polling attempts for each UNIQUE GBIF download key.

Details

`keys` can be a named list or named character vector. Names represent logical request labels such as species names or ISO2 countries, while values are GBIF asynchronous download keys.

Multiple labels are allowed to share the same GBIF key. Before polling, biofetchR constructs the request-label-to-key mapping and reduces the supplied keys to their unique non-missing values. Each unique GBIF download is therefore polled, downloaded and imported only once.

For each unique key, GBIF metadata are polled up to `max_tries` times with `wait_time` seconds between attempts. `SUCCEEDED` proceeds to import. `FAILED`, `KILLED`, `CANCELLED`, and `FILE_ERASED` are treated as terminal failure states. If no terminal state is reached before the polling limit, the request is recorded as `pending_timeout`; the GBIF key is retained for later review or recovery.

Completed archives are imported with `rgbif::occ_download_import()`. Records lacking `decimalLatitude` or `decimalLongitude` are removed before conversion to EPSG:4326 point geometry.

Value

A named list of WGS84 `sf` point objects for successfully resolved request labels. For a normal species-level download, one imported object is returned under the corresponding species label. For a shared multi-country download, the archive is imported once and separate country-labelled `sf` objects are returned after splitting by the configured field.

Successfully resolved labels with zero matching records are retained as zero-row `sf` objects. Failed or unresolved requests need not appear as list elements, but every original request is represented in the attached `gbif_status` audit attribute.

Shared download keys and country splitting

The terrestrial/freshwater multi-country backend deliberately returns several country labels pointing to one shared GBIF download key and attaches `biofetchR_split_field = "countryCode"`.

When a split field is present, the completed archive is imported once and then partitioned locally for each original label. A country with no matching rows after a successful import is retained as a zero-row WGS84 `sf` object and is audited as `success_zero`, rather than disappearing from the returned result.

If no split field is supplied, each request label associated with a successful key receives the complete imported `sf` object.

Retrieval-status audit

The returned list carries a `gbif_status` attribute containing one row per original request label. Its columns are:

- `label`: original request label;
- `gbif_key`: associated GBIF download key;
- `gbif_status`: terminal or last observed GBIF download status;
- `retrieval_status`: biofetchR retrieval interpretation;
- `n_records`: number of coordinate-bearing records assigned to the request label when known;
- `message`: diagnostic or recovery message where relevant.

Retrieval statuses include `success`, `success_zero`, `invalid_key`, `pending_timeout`, `download_failed`, `killed`, `cancelled`, `file_erased`, `import_failed`, and `split_failed`.

Importantly, `pending_timeout` means that the download remained unresolved within the configured polling window. It is not interpreted as a biological zero and the associated GBIF key remains available in the audit.

GBIF data use and citation

This function imports GBIF occurrence-download data. Retain the GBIF download key/DOI in downstream summaries and cite the corresponding GBIF occurrence download, or an appropriate derived dataset, when using the records in analyses or publications.

See Also

Other GBIF helpers: `check_gbif_presence()`, `get_taxon_key()`

Examples

```
if (interactive() && exists("keys")) {
  imported <- wait_and_import_gbif(keys)

  names(imported)
  attr(imported, "gbif_status")
}
```

`wait_and_import_gbif_safe` *Import GBIF downloads through the package import helper*

Description

Compatibility wrapper around `wait_and_import_gbif()`. It exposes a safer, explicit function name expected by some package scripts while preserving the existing import behaviour implemented elsewhere in biofetchR.

Usage

```
wait_and_import_gbif_safe(download_keys, ...)
```

Arguments

`download_keys`	GBIF download keys, usually returned by `download_gbif_batch()` or `download_gbif_batch_gadm()`. The accepted object shape is whatever `wait_and_import_gbif()` supports.
`...`	Additional arguments passed directly to `wait_and_import_gbif()`, such as polling or retry settings if supported by that function.

Details

This wrapper is intentionally minimal: it checks that `wait_and_import_gbif()` is visible, then forwards `download_keys` and any additional arguments to that function.

Value

The object returned by `wait_and_import_gbif()`. In normal biofetchR workflows this is a named list of imported GBIF occurrence tables or `sf` point objects, with names corresponding to the supplied download-key labels. The exact structure depends on the available `wait_and_import_gbif()` implementation. This wrapper does not alter the imported records; it is called primarily to provide a stable package-visible import helper name for pipelines and tests.

GBIF data use and citation

The imported records should remain traceable to the GBIF download keys and associated GBIF download DOI(s). Downstream workflows should preserve these keys in manifests or summaries so users can cite the exact occurrence downloads used in analysis.

Errors

Stops if `wait_and_import_gbif()` is not available in the loaded package namespace or search path.

See Also

download_gbif_batch(), download_gbif_batch_gadm()

Other GBIF download backends: download_gbif_batch(), download_gbif_batch_gadm()

Examples

```
if (interactive() && exists("keys")) {
  imported <- wait_and_import_gbif_safe(download_keys = keys)
}
```

Index