

Package: taxifydb (via r-universe)

October 12, 2026

Title Build Backbones and Enrichments for the 'taxify' Package

Version 0.1.23.9000

Description Build pipeline for the 'taxify' package. Downloads raw source data from official providers (WFO, COL, GBIF, ITIS, NCBI Taxonomy, Open Tree of Life, WoRMS, Euro+Med PlantBase, Index Fungorum, AlgaeBase, FishBase, SeaLifeBase, the Reptile Database, the World Checklist of Vascular Plants, and the Leipzig Catalogue of Vascular Plants) and a wide set of trait and conservation datasets, normalizes to a unified Darwin Core-like schema, and writes pre-compiled '.vtr' files that the 'taxify' runtime consumes. Separates build-time concerns (network access, parsing, schema normalization) from runtime concerns so that 'taxify' itself stays lean.

License MIT + file LICENSE

URL <https://github.com/gcol33/taxifydb>

BugReports <https://github.com/gcol33/taxifydb/issues>

Depends R (>= 4.1.0)

Imports curl, digest, jsonlite, taxify (>= 0.6.0.9000), utils, vectra (>= 0.12.4)

Suggests data.table, DBI, GIFT, openxlsx2, readxl, rfishbase, rvest, RSQLite, testthat (>= 3.2.0)

Remotes gcol33/taxify

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libssl-dev

Repository <https://gcol33.r-universe.dev>

Date/Publication 2026-10-11 23:55:23 UTC

RemoteUrl <https://github.com/gcol33/taxifydb>

RemoteRef HEAD

RemoteSha 540e1e9b525b242b705cc9c27a600f26835e6cde

Contents

apply_delta	8
assert_built_matches_registry	8
attach_references	9
build_algaebase	9
build_all_name_lookups	10
build_avilist	10
build_backend	11
build_backend_coverage	11
build_betsi_recovery	12
build_col	13
build_colxr	13
build_enrichment	14
build_enrichment_vtr	15
build_euromed	17
build_evc	17
build_fishbase	18
build_fungorum	18
build_gbif	19
build_gbif_occurrence_counts	19
build_genus_register	20
build_itis	21
build_lcvp	21
build_lpsn	22
build_mdd	22
build_meow	23
build_name_lookup	23
build_ncbi	24
build_ott	24
build_register	25
build_reptiledb	26
build_sealifebase	26
build_vegcz	27
build_vtr	27
build_vtr_streamed	28
build_wcvp	30
build_wfo	30
build_wgsrpd	31
build_worms	31
check_all_enrichment_versions	32
check_ckan_version	32
check_content_md5	33
check_crossref_version	33

check_datacite_version	34
check_dataone_version	35
check_dataverse_version	35
check_dryad_version	36
check_edi_version	36
check_enrichment_source_version	37
check_figshare_version	38
check_gbif_api_version	38
check_gift_version	39
check_github_version	39
check_ipt_version	40
check_kew_cvalues_version	40
check_last_modified	41
check_lcvp_version	41
check_ncbi_archive_version	42
check_ott_version	42
check_pangaea_version	43
check_phaidra_version	43
check_rfishbase_version	44
check_seanoe_version	44
check_wayback_version	45
check_wst_version	45
check_zenodo_version	46
colxr_col_publications	47
colxr_gbif_crosswalk	47
colxr_gbif_lookup	48
colxr_latest_release	48
colxr_request_export	49
count_gbif_occurrences	50
count_vtr_rows	50
create_delta	51
delim_chunk_feed	51
delim_fk_lookup	53
download_algaebase	54
download_and_unzip	54
download_avilist	55
download_cf_file	55
download_col	56
download_colxr	56
download_curl_file	57
download_dryad_file	58
download_edi_file	58
download_euromed	59
download_evc	60
download_fungorum	60
download_gbif	61
download_gbif_api_pages	61
download_gbif_occurrence_counts	62

download_groot	62
download_itis	63
download_lcvp	63
download_lpsn	64
download_mdd	64
download_meow	65
download_ncbi	65
download_ott	66
download_reptiledb	66
download_vegcz	67
download_wcvp	67
download_wfo	68
download_wgsrpd	68
download_wiley_supplement	69
download_worms	69
enrichment_emergency_fallback	70
enrichment_trait_cols	70
gbif_count_keys	71
gen_spe	71
gunzip_file	72
harvest_ckan_datastore	72
harvest_plazi_dwca	73
has_xdelta3	73
inrae_genus_dict	74
list_backends	74
list_betsi_recovery	75
list_build_only_enrichments	75
list_enrichments	76
list_geometry	76
list_scrape_only_enrichments	76
normalize_backbone	77
normalize_colxr	77
normalize_gbif	78
normalize_wcvp	79
normalize_wfo	79
normalize_worms	80
parse_algae_traits	80
parse_alien_first_records	81
parse_alpine_carabids	82
parse_amniote	82
parse_amphibio	83
parse_anage	83
parse_animaltraits	84
parse_arctic	84
parse_arthropod_traits	85
parse_austraits	85
parse_avonet	86
parse_bacdiv	86

parse_baseflor	87
parse_bee_ostwald	87
parse_bet	88
parse_betsi_collembola_body_length	88
parse_betsi_recovery	89
parse_beukhof	89
parse_bien	90
parse_birdbase	90
parse_blanchard	91
parse_brot	91
parse_bryoatt	92
parse_ccdb	92
parse_cefas_btrait	93
parse_chelonians	93
parse_chowdhury	94
parse_clopla	94
parse_combine	95
parse_common_names	95
parse_copepod_traits	96
parse_coral_traits	97
parse_diaz_traits	97
parse_disperse	98
parse_eberswalde	98
parse_ecoflora	99
parse_ecomorphosis	100
parse_edwards_phyto	100
parse_eive	101
parse_ellers_collembola	101
parse_elton_traits	102
parse_epa_freshwater	103
parse_eupolltrait	103
parse_eurobat	104
parse_euromed_distribution	104
parse_faprotax	105
parse_finand	106
parse_fishbase	106
parse_fishmorph	107
parse_fishtraits	107
parse_floraweb	108
parse_frugivoria	109
parse_fungal_traits	109
parse_fungalroot	110
parse_funguild	110
parse_fw_insects_conus	111
parse_gbif_common_names	111
parse_gidias	112
parse_gift	113
parse_globi	114

parse_globtherm	114
parse_glonaf	115
parse_gmpd	115
parse_griis	116
parse_groot	116
parse_gwdd	117
parse_homerange	117
parse_hosts	118
parse_huang_amph	118
parse_hydraulics	119
parse_imageomics_neon	120
parse_invacost	120
parse_italic	121
parse_iucn	122
parse_kew_cvalues	122
parse_kew_sid	123
parse_leda	124
parse_leptrait	125
parse_madin	125
parse_marine_distribution	126
parse_monograph_body_length	126
parse_monograph_collembola	127
parse_ncbi_common_names	128
parse_nesttrait	128
parse_noddb	129
parse_nztd	129
parse_octocoral	130
parse_odonata	130
parse_ott_common_names	131
parse_pantheria	131
parse_parravicini	132
parse_pelagic	132
parse_phylacine	133
parse_plantatt	133
parse_plazi_collembola_body_length	134
parse_pottier	134
parse_quimbayo	135
parse_ramond	135
parse_repttraits	136
parse_rimet_phyto	136
parse_saproxylic	137
parse_sealifebase	137
parse_sharkipedia	138
parse_sheld	138
parse_spider_traits	139
parse_sworm	139
parse_tetradensity	140
parse_thermofresh	140

parse_tree_of_sex	141
parse_usda_fungus_host	142
parse_useful_plants	142
parse_virion	143
parse_wcvp	143
parse_zanne	144
parse_zooplankton	144
precompute_backbone	145
probe_upstream_identity	145
publish_enrichment_release	146
publish_release	147
read_algaebase	148
read_avilist	148
read_col	149
read_colxr	149
read_dmp	150
read_euromed	150
read_evc	151
read_fungorum	151
read_gbif	152
read_gbif_occurrence_counts	152
read_itis	153
read_lcvp	153
read_lpsn	154
read_mdd	154
read_meow	155
read_ncbi	155
read_ott	156
read_reptiledb	156
read_vegcz	157
read_wcvp	157
read_wfo	158
read_wgsrpd	158
read_worms	159
register_backbones	159
resolve_betsi_codes	160
resolve_enrichment_names	161
resolve_hierarchy	162
resolve_name_map	163
rfishbase_release	164
sha256	165
split_scientific_name	165
tsita_crosswalk	166
tsita_vocab	167
update_enrichment_manifest	167
update_manifest	168
vtr_changed	170
wfo_latest_edition	170

apply_delta	<i>Apply a binary delta to produce a new .vtr</i>
-------------	---

Description

Apply a binary delta to produce a new .vtr

Usage

apply_delta(old_path, delta_path, new_path)

Arguments

- | | |
|------------|---|
| old_path | Character. Path to the current local .vtr. |
| delta_path | Character. Path to the downloaded .xdelta file. |
| new_path | Character. Output path for the patched .vtr. |

Value

The new path (invisibly), or NULL on failure.

assert_built_matches_registry	<i>Check that an already-built enrichment matches its registry entry</i>
-------------------------------	--

Description

TAXIFYDB_PUBLISH_RESUME=1 reuses whatever .vtr sits in an enrichment’s output directory instead of rebuilding it, which is what lets an interrupted publish finish cheaply. It also means a .vtr left from an earlier source release looks exactly like a current one: publishing it uploads stale data and writes a manifest entry naming a source that build never read. The meta.json sidecar records the URL and version the build actually used, so that is what the registry is compared against.

Usage

assert_built_matches_registry(name, vtr_path)

Arguments

- | | |
|----------|---|
| name | Character. Enrichment name in the build registry. |
| vtr_path | Character. Path to the built .vtr; its meta.json sidecar is read from the same directory. |

Value

TRUE, invisibly. Stops when the build and the registry disagree.

attach_references	<i>Attach a reference table to a parsed enrichment</i>
-------------------	--

Description

Marks df as carrying per-value provenance: its <col>_source columns hold reference ids that resolve against references. `build_enrichment()` carries the table through name resolution and `build_enrichment_vtr()` publishes it.

Usage

```
attach_references(df, references)
```

Arguments

df	The parser's data.frame, keyed on canonical_name.
references	data.frame with a unique character ref_id, a citation (NA for an id the source names but gives no citation for) and a doi (NA where the reference has none). Further columns are kept.

Value

df, with the table in its references attribute.

build_algaebase	<i>Build the AlgaeBase backbone .vtr</i>
-----------------	--

Description

Build the AlgaeBase backbone .vtr

Usage

```
build_algaebase(
  output_dir = "output/algaebase",
  version = NULL,
  verbose = TRUE
)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_all_name_lookups

Build name-lookup tables for all installed taxify backbones

Description

Locates each backbone in the user's taxify data dir and writes a {backend}_name_lookup.vtr alongside it.

Usage

```
build_all_name_lookups(backends = list_backends(), overwrite = FALSE)
```

Arguments

backends	Character vector. Default: every taxify backbone (list_backends()).
overwrite	Logical. Rebuild even if the lookup .vtr already exists.

Value

Character vector of paths to the built lookups.

build_avilist

Build the AviList backbone .vtr from source

Description

Build the AviList backbone .vtr from source

Usage

```
build_avilist(output_dir = "output/avilist", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. Defaults to the bundled AviList version.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_backend	<i>Build a taxify backbone .vtr file from source</i>
---------------	--

Description

Single entry point for backbone builds. Downloads the raw source, normalizes to the unified schema, precomputes matching keys, and writes the final .vtr with indexes and metadata sidecar.

Usage

```
build_backend(name, output_dir = NULL, version = NULL, verbose = TRUE)
```

Arguments

name	Character. Backend identifier (e.g., "itis"). See list_backends() for the taxonomic backbones and list_geometry() for the reference-geometry artifacts; both are buildable here.
output_dir	Character. Output directory. Default: output/<name>.
version	Character or NULL. Version string for the build. If NULL, defaults to the current YYYY.MM (or the backend's bundled default).
verbose	Logical.

Value

Path to the built .vtr file (invisibly).

build_backend_coverage	<i>Build the backend coverage table</i>
------------------------	---

Description

For each backbone in [register_backbones\(\)](#) (resolved via [resolve_register_backbone_paths\(\)](#)), extracts the genus list and writes a long-format backend_coverage.vtr recording which genera are covered by which backend and at what version/date.

Usage

```
build_backend_coverage(
  backbone_paths = NULL,
  output_dir = NULL,
  version = NULL,
  manifest_path = "manifest/manifest.json",
  verbose = TRUE
)
```

Arguments

backbone_paths	Named list/character vector or NULL. Explicit .vtr path overrides for one or more of register_backbones() . Backbones not named here resolve from output/<name>/<name>.vtr (a prior build_backend() run) or from the version published in manifest.json.
output_dir	Character or NULL. Output directory. Default output/backend_coverage.
version	Character or NULL. Version string for the build. If NULL, defaults to the current YYYY.MM.
manifest_path	Character. Path to manifest.json, used to resolve backbones not present locally.
verbose	Logical. Print progress messages. Default TRUE.

Value

Path to backend_coverage.vtr (invisibly).

build_betsi_recovery *Build a BETSI-recovery enrichment .vtr from its frozen matrices*

Description

Convenience wrapper over [build_enrichment\(\)](#) restricted to the per-taxon assets rebuilt from published BETSI-derived matrices (see [list_betsi_recovery\(\)](#)). The heavy lifting – staging the frozen matrices, the parse, cross-backbone name resolution, T-SITA + provenance metadata, the .vtr write – is the shared enrichment pipeline; this only guards the name.

Usage

```
build_betsi_recovery(name, ...)
```

Arguments

name	Character. A recovery enrichment name.
...	Passed to build_enrichment() (output_dir, version, ...).

Value

Path to the built .vtr (invisibly).

See Also

[list_betsi_recovery\(\)](#)

build_col	<i>Build the COL backbone .vtr</i>
-----------	------------------------------------

Description

Also writes the SpeciesProfile.tsv (extinct/marine flags) as a separate col_species_profile.vtr next to the main backbone, if present.

Usage

```
build_col(output_dir = "output/col", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_colxr	<i>Build the COL XR backbone .vtr</i>
-------------	---------------------------------------

Description

Build the COL XR backbone .vtr

Usage

```
build_colxr(  
  output_dir = "output/colxr",  
  version = NULL,  
  gbif_path = NULL,  
  col_path = NULL,  
  manifest_path = "manifest/manifest.json",  
  verbose = TRUE  
)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. The COL XR release version, resolved from ChecklistBank when NULL so the stamped version is the date of the data rather than the date of the build.
gbif_path	Character or NULL. The gbif .vtr the gbif_key column is derived from. Defaults to the published build named in manifest_path, else output/gbif/gbif.vtr.
col_path	Character or NULL. The col .vtr the publication references are read from. Defaults to the published build named in manifest_path, else output/col/col.vtr.
manifest_path	Character. Path to manifest.json.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_enrichment	<i>Build an enrichment .vtr from source</i>
------------------	---

Description

Downloads the raw source for an enrichment, parses it via the registered parser, expands names across all 7 taxify backbones via `resolve_enrichment_names()`, and writes a .vtr plus meta.json sidecar via `build_enrichment_vtr()`.

Usage

```
build_enrichment(  
  name,  
  output_dir = NULL,  
  version = NULL,  
  url = NULL,  
  resolve_names = TRUE,  
  strict_names = FALSE,  
  verbose = TRUE  
)
```

Arguments

name	Character. Enrichment identifier. See <code>list_enrichments()</code> .
output_dir	Character or NULL. Output directory. Default output/enrichment/<name>.
version	Character or NULL. Version override. If NULL, uses the registry's pinned version.
url	Character or NULL. Custom source URL override. If supplied, the build version defaults to <code>format(Sys.Date(), "%Y.%m")</code> and source_doi is set to NULL.

resolve_names	Logical. Run <code>resolve_enrichment_names()</code> before writing. Default TRUE.
strict_names	Logical. Error when a backbone has no <code>name_lookup.vtr</code> , instead of expanding against a narrower set and warning. FALSE by default so a user building an enrichment locally against whatever they have installed still works; the asset-publishing pipeline sets TRUE, since a published asset must not be silently expanded against a partial backbone set.
verbose	Logical. Default TRUE.

Value

Path to the built `.vtr` file (invisibly).

build_enrichment_vtr	<i>Write an enrichment .vtr file</i>
----------------------	--------------------------------------

Description

Sorts by `canonical_name`, writes the `.vtr`, creates hash indexes on `canonical_name` (and optionally a group column), and writes a `meta.json` sidecar.

Usage

```
build_enrichment_vtr(
  df,
  vtr_path,
  name,
  version,
  source_url,
  source_doi = NULL,
  source_record = NULL,
  upstream_id = NULL,
  license = "unknown",
  attribution = NULL,
  group_col = NULL,
  species_col = NULL,
  static = TRUE,
  source_format = NULL,
  provenance = NULL,
  references = NULL,
  resolved_backbones = NULL,
  batch_size = 50000L
)
```

Arguments

df	A data.frame with at least a canonical_name column.
vtr_path	Character. Output path for the .vtr file.
name	Character. Enrichment identifier (e.g., "zanne").
version	Character. Version string (e.g., "2026.04").
source_url	Character. URL the source data was downloaded from.
source_doi	Character or NULL. DOI of the source dataset.
source_record	Character or NULL. The host record a download URL belongs to, where neither the URL nor source_doi names it (a Figshare file URL beside a journal DOI). Read by check_figshare_version() .
upstream_id	Character or NULL. The identity the source host gives the version this build read – a Zenodo record number, a Figshare or Dryad version number, a Last-Modified stamp. Recorded so the weekly freshness check compares upstream against upstream; see probe_upstream_identity() .
license	Character. License string (e.g., "CC0", "CC BY 4.0").
attribution	Character. Human-readable attribution string.
group_col	Character or NULL. Column to index for group-based enrichments (e.g., "country_code", "tdwg_code", "lang").
species_col	Character or NULL. Name of the source column the taxa were keyed on, recorded for the runtime manifest. "genus" marks a genus-grain enrichment; NULL (the default) is species-grain.
static	Logical. Whether the enrichment is a frozen snapshot (the default) rather than one the runtime re-checks against a live source. Drives taxify's refresh gate (content-id for static, version for non-static).
source_format	Character or NULL. Raw source format (e.g. "csv", "xlsx", "zip"), recorded for the runtime manifest.
provenance	Named list or NULL. Per-column provenance: maps each trait column to its provenance tier, recorded in meta.json. Used by assets whose columns come from mixed sources (the BETSI-recovery matrices); NULL (the default) writes no provenance block.
references	data.frame or NULL. The reference table behind the enrichment's <col>_source provenance columns (see attach_references()). Written as <name>_references.vtr beside vtr_path, keeping the references some cell names, and declared in meta.json as references (file, nrow, content_id). Every id a provenance cell names must be in the table. NULL (the default) writes none.
resolved_backbones	Character vector or NULL. The backbones the cross-backbone name expansion actually reached. Recorded so an asset built against a partial backbone set is distinguishable afterwards from a complete one, which is what a bare warning at build time did not achieve.
batch_size	Integer. Row group size for vectra (default 50000).

Value

The path to the .vtr file (invisibly).

build_euromed	<i>Build the Euro+Med backbone .vtr</i>
---------------	---

Description

Build the Euro+Med backbone .vtr

Usage

```
build_euromed(  
  output_dir = "output/euromed",  
  version = NULL,  
  local_dir = NULL,  
  verbose = TRUE  
)
```

Arguments

output_dir	Character.
version	Character or NULL.
local_dir	Character or NULL. Local crawl snapshot directory (see download_euromed()).
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_evc	<i>Build the EuroVegChecklist syntaxon backbone .vtr</i>
-----------	--

Description

Build the EuroVegChecklist syntaxon backbone .vtr

Usage

```
build_evc(  
  output_dir = "output/evc",  
  version = NULL,  
  local_dir = NULL,  
  verbose = TRUE  
)
```

Arguments

output_dir	Character.
version	Character or NULL. The EuroVegChecklist version the unit list carries (default the snapshot's).
local_dir	Character or NULL. Local crawl snapshot directory (see download_evc()).
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_fishbase	<i>Build the FishBase backbone .vtr</i>
----------------	---

Description

Build the FishBase backbone .vtr

Usage

```
build_fishbase(output_dir = "output/fishbase", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_fungorum	<i>Build the Species Fungorum Plus backbone .vtr</i>
----------------	--

Description

Build the Species Fungorum Plus backbone .vtr

Usage

```
build_fungorum(output_dir = "output/fungorum", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_gbif	<i>Build the GBIF backbone .vtr</i>
------------	-------------------------------------

Description

Build the GBIF backbone .vtr

Usage

```
build_gbif(output_dir = "output/gbif", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_gbif_occurrence_counts	<i>Build the GBIF occurrence-count snapshot</i>
------------------------------	---

Description

Collects the keys `gbif_count_keys()` selects from a built GBIF backbone, counts their occurrence records, and writes them as a gzipped TSV for release under gbif-occurrence-counts-<version>.

Usage

```
build_gbif_occurrence_counts(vtr_path, out_dir = "output/gbif", verbose = TRUE)
```

Arguments

vtr_path	Character. Path to a built GBIF backbone .vtr.
out_dir	Character. Output directory.
verbose	Logical.

Value

Path to the written .tsv.gz (invisibly).

build_genus_register *Build the genus register from a fixed backbone set*

Description

Reads genus-rank rows from each of `register_backbones()` (resolved via `resolve_register_backbone_paths()`), unions them, resolves classification conflicts, normalizes non-standard kingdom names, assigns kingdom_group/taxon_group/life_form (via `assign_life_form()`), and writes `genus_register.vtr`.

Usage

```
build_genus_register(
  backbone_paths = NULL,
  output_dir = NULL,
  version = NULL,
  manifest_path = "manifest/manifest.json",
  verbose = TRUE
)
```

Arguments

backbone_paths	Named list/character vector or NULL. Explicit .vtr path overrides for one or more of <code>register_backbones()</code> . Backbones not named here resolve from <code>output/<name>/<name>.vtr</code> (a prior <code>build_backend()</code> run) or from the version published in <code>manifest.json</code> .
output_dir	Character or NULL. Output directory. Default <code>output/genus_register</code> .
version	Character or NULL. Version string for the build. If NULL, defaults to the current YYYY.MM.
manifest_path	Character. Path to <code>manifest.json</code> , used to resolve backbones not present locally.
verbose	Logical. Print progress messages. Default TRUE.

Details

Every call unions the same fixed backbone set, so the built register is reproducible: it does not depend on which backbones happen to be installed on the machine running the build (contrast `taxify`'s runtime `taxify_build_register()`, which unions whichever backbones the caller has installed).

Value

Path to genus_register.vtr (invisibly).

build_itis	<i>Build the ITIS backbone .vtr from scratch</i>
------------	--

Description

Downloads the ITIS SQLite dump, normalizes it, precomputes matching keys, embeds synonym info, and writes the final .vtr with indexes.

Usage

```
build_itis(output_dir = "output/itis", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Directory for the output .vtr.
version	Character or NULL. If NULL, the YYYY.MM of the release the ITIS download currently holds.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_lcvp	<i>Build the LCVP backbone .vtr from source</i>
------------	---

Description

Build the LCVP backbone .vtr from source

Usage

```
build_lcvp(output_dir = "output/lcvp", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. Defaults to the tagged LCVP release the download is pinned to.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_lpsn	<i>Build the LPSN backbone .vtr from source</i>
------------	---

Description

Build the LPSN backbone .vtr from source

Usage

```
build_lpsn(output_dir = "output/lpsn", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. Defaults to the release ChecklistBank currently serves.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_mdd	<i>Build the Mammal Diversity Database backbone .vtr from source</i>
-----------	--

Description

Build the Mammal Diversity Database backbone .vtr from source

Usage

```
build_mdd(output_dir = "output/mdd", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. Defaults to the MDD version the downloaded archive names in its species file (MDD_v2.5_...species.csv).
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_meow	<i>Build the MEOW ecoregion boundary .vtr</i>
------------	---

Description

Downloads the MEOW ecoregion GeoJSON snapshot (or reads a local copy), parses it to a long-format vertex table, and writes meow.vtr with an index on code and a metadata sidecar. taxify downloads this pre-built .vtr and rebuilds the boundary geometry offline, so the runtime never fetches the GeoJSON. The marine analogue of [build_wgsrpd\(\)](#).

Usage

```
build_meow(
  output_dir = NULL,
  version = NULL,
  source_path = NULL,
  verbose = TRUE
)
```

Arguments

output_dir	Character. Output directory. Default: output/meow.
version	Character or NULL. Version string. Defaults to the frozen snapshot's version.
source_path	Character or NULL. A local GeoJSON to parse instead of downloading (for offline builds and testing).
verbose	Logical.

Value

Path to the built .vtr (invisibly).

build_name_lookup	<i>Build a name-lookup .vtr from a backbone .vtr</i>
-------------------	--

Description

Build a name-lookup .vtr from a backbone .vtr

Usage

```
build_name_lookup(bb_path, out_path, verbose = TRUE)
```

Arguments

bb_path	Character. Backbone .vtr path.
out_path	Character. Lookup .vtr destination.
verbose	Logical.

Value

out_path (invisibly).

build_ncbi	<i>Build the NCBI backbone .vtr</i>
------------	-------------------------------------

Description

Build the NCBI backbone .vtr

Usage

```
build_ncbi(output_dir = "output/ncbi", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_ott	<i>Build the OTT backbone .vtr</i>
-----------	------------------------------------

Description

Build the OTT backbone .vtr

Usage

```
build_ott(output_dir = "output/ott", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_register	<i>Build both genus_register.vtr and backend_coverage.vtr</i>
----------------	---

Description

Convenience wrapper that calls `build_genus_register()` and `build_backend_coverage()` in sequence. Resolves every backbone's .vtr exactly once (via `resolve_register_backbone_paths()`, downloading into `output/genus_register/_cache` when needed) and passes the fully resolved paths to both builders as `backbone_paths`, so a manifest-downloaded backbone (several hundred MB each, ~5 GB across the set) is fetched once for both artifacts rather than once per builder.

Usage

```
build_register(
  backbone_paths = NULL,
  version = NULL,
  manifest_path = "manifest/manifest.json",
  verbose = TRUE
)
```

Arguments

<code>backbone_paths</code>	Named list/character vector or NULL. Explicit .vtr path overrides for one or more of <code>register_backbones()</code> . Backbones not named here resolve from <code>output/<name>/<name>.vtr</code> (a prior <code>build_backend()</code> run) or from the version published in <code>manifest.json</code> .
<code>version</code>	Character or NULL. Version string for the build. If NULL, defaults to the current YYYY.MM.
<code>manifest_path</code>	Character. Path to <code>manifest.json</code> , used to resolve backbones not present locally.
<code>verbose</code>	Logical. Print progress messages. Default TRUE.

Value

Named list with paths to `genus_register.vtr` and `backend_coverage.vtr` (invisibly).

build_reptiledb	<i>Build the Reptile Database backbone .vtr</i>
-----------------	---

Description

Build the Reptile Database backbone .vtr

Usage

```
build_reptiledb(  
    output_dir = "output/reptiledb",  
    version = NULL,  
    verbose = TRUE  
)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_sealifebase	<i>Build the SeaLifeBase backbone .vtr</i>
-------------------	--

Description

Build the SeaLifeBase backbone .vtr

Usage

```
build_sealifebase(  
    output_dir = "output/sealifebase",  
    version = NULL,  
    verbose = TRUE  
)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_vegcz	<i>Build the Czech vegetation classification syntaxon backbone .vtr</i>
-------------	---

Description

Build the Czech vegetation classification syntaxon backbone .vtr

Usage

```
build_vegcz(  
  output_dir = "output/vegcz",  
  version = NULL,  
  local_dir = NULL,  
  verbose = TRUE  
)
```

Arguments

- output_dir Character.
- version Character or NULL. Default the snapshot's.
- local_dir Character or NULL. Local crawl snapshot directory (see [download_vegcz\(\)](#)).
- verbose Logical.

Value

Path to the .vtr file (invisibly).

build_vtr	<i>Write a backbone .vtr file from a precomputed data.frame</i>
-----------	---

Description

Sorts by genus (for zone-map pruning), writes the .vtr, creates hash indexes, and writes a metadata sidecar.

Usage

```
build_vtr(
  df,
  vtr_path,
  backend_name,
  version,
  source_url,
  source_date,
  batch_size = 50000L,
  block_col = NULL,
  extra = NULL
)
```

Arguments

df	A precomputed backbone data.frame (from <code>precompute_backbone()</code>).
vtr_path	Character. Output path for the .vtr file.
backend_name	Character. Backend identifier (e.g., "itis").
version	Character. Version string.
source_url	Character. URL the source data was downloaded from.
source_date	Character. The date the source gives the release it was built from (YYYY-MM-DD, or YYYY-MM / YYYY where the source dates it no finer), recorded in the .meta sidecar and from there in the manifest.
batch_size	Integer. Row group size for vectra (default 50000).
block_col	Character or NULL. The column the fuzzy pass blocks candidates by, which the store is sorted and indexed on: NULL takes the genus column of a taxon backbone; a syntaxon backbone passes "block_key".
extra	Named character vector of further .meta fields, or NULL.

Value

The path to the .vtr file (invisibly).

build_vtr_streamed	<i>Build a backbone .vtr from a stream of normalized chunks</i>
--------------------	---

Description

The streaming counterpart to `build_vtr()`, for a source too large to assemble in memory. Rows arrive a chunk at a time from feed; the finished store is identical to what `build_vtr()` would have written from the concatenation of those chunks.

Usage

```
build_vtr_streamed(
  feed,
  vtr_path,
  backend_name,
  version,
  source_url,
  source_date,
  synonym_pattern = "SYNONYM",
  meta_extra = NULL,
  batch_size = 50000L,
  work_dir = NULL,
  verbose = TRUE
)
```

Arguments

feed	A function of no arguments returning the next chunk of rows as a normalized data.frame (the schema normalize_backbone() produces), or NULL when the source is exhausted. A zero-row data.frame means the block was filtered out entirely, not that the source has ended.
vtr_path	Character. Output path for the .vtr file.
backend_name	Character. Backend identifier (e.g. "colxr").
version	Character. Version string.
source_url	Character. URL the source data was downloaded from.
source_date	Character. The source's release date, as for build_vtr() .
synonym_pattern	Character. Regex matching the statuses that count as a synonym, as passed to precompute_backbone() .
meta_extra	Named character vector of further key=value lines for the .meta sidecar (e.g. the input snapshot a column was read from), or NULL.
batch_size	Integer. Row group size for vectra (default 50000).
work_dir	Character. Directory for the staging store. Defaults to a session temporary directory, removed on exit.
verbose	Logical.

Value

The path to the .vtr file (invisibly).

build_wcvp	<i>Build the WCVF backbone .vtr from source</i>
------------	---

Description

Build the WCVF backbone .vtr from source

Usage

```
build_wcvp(output_dir = "output/wcvp", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. Defaults to the bundled WCVF snapshot tag.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_wfo	<i>Build the WFO backbone .vtr from source</i>
-----------	--

Description

Build the WFO backbone .vtr from source

Usage

```
build_wfo(output_dir = "output/wfo", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character. Output directory.
version	Character or NULL. Defaults to the edition wfo_latest_edition() resolves.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

build_wgsrpd	<i>Build the WGSRPD Level 3 boundary .vtr</i>
--------------	---

Description

Downloads the WGSRPD Level 3 GeoJSON (or reads a local copy), parses it to a long-format vertex table, and writes wgsrpd.vtr with an index on code and a metadata sidecar. taxify downloads this pre-built .vtr and rebuilds the boundary geometry offline, so the runtime never fetches the GeoJSON.

Usage

```
build_wgsrpd(
  output_dir = NULL,
  version = NULL,
  source_path = NULL,
  verbose = TRUE
)
```

Arguments

output_dir	Character. Output directory. Default: output/wgsrpd.
version	Character or NULL. Version string. Defaults to the current YYYY.MM.
source_path	Character or NULL. A local GeoJSON to parse instead of downloading (for offline builds and testing).
verbose	Logical.

Value

Path to the built .vtr (invisibly).

build_worms	<i>Build the WoRMS backbone .vtr</i>
-------------	--------------------------------------

Description

Also writes the SpeciesProfile.tsv (habitat flags) as a separate worms_species_profile.vtr next to the main backbone.

Usage

```
build_worms(output_dir = "output/worms", version = NULL, verbose = TRUE)
```

Arguments

output_dir	Character.
version	Character or NULL.
verbose	Logical.

Value

Path to the .vtr file (invisibly).

check_all_enrichment_versions

Check all non-static enrichments in a manifest for version freshness

Description

Check all non-static enrichments in a manifest for version freshness

Usage

```
check_all_enrichment_versions(manifest_path = "manifest/manifest.json")
```

Arguments

manifest_path	Character. Path to manifest.json.
---------------	-----------------------------------

Value

Data.frame with columns: name, source_version, built_id, upstream_id, upstream_version, out-dated, check_url, note.

check_ckan_version

Check a CKAN resource for its last modification

Description

The resource is named by /resource/<id> in the URL, else in the source's host record (a portal whose downloads go through its datastore API).

Usage

```
check_ckan_version(source_url, record = NULL)
```

Arguments

source_url	Character. CKAN resource download URL.
record	Character or NULL. CKAN resource URL, when source_url does not carry one.

Value

Named list with id (the resource's last_modified stamp), version, date, and url; NULL if the resource cannot be read.

check_content_md5	<i>Check a file by the md5 of its bytes</i>
-------------------	---

Description

For a host that serves a generated export with no version and no Last-Modified header (a PHP endpoint, a database export). The bytes are the only identity there is, so the probe downloads them; it is listed only for sources small enough to fetch weekly, and only where two downloads hash alike.

Usage

```
check_content_md5(source_url, record = NULL)
```

Arguments

source_url	Character. File URL.
record	Unused.

Value

Named list with id (the md5), version (its first 8 characters), and url; NULL if the file cannot be read.

check_crossref_version	<i>Check a journal article for corrections</i>
------------------------	--

Description

A published article does not get new versions, it gets corrections and retractions, which Crossref records against it as updated-by. A source whose values come from an article is therefore checked by that list.

Usage

```
check_crossref_version(source_url, record = NULL)
```

Arguments

source_url	Character. doi.org URL of the article, or a URL that is not one, in which case the DOI comes from record.
record	Character or NULL. The article DOI.

Value

Named list with id (the updating DOIs, or "none"), version, date (the latest of publication and update), and url; NULL if Crossref does not know the DOI.

check_datacite_version	<i>Check a dataset DOI for the version DataCite records</i>
------------------------	---

Description

For a host whose own pages give no version but whose DataCite record carries one. The record's updated stamp moves with any metadata edit, so as a date it errs late, never early.

Usage

```
check_datacite_version(source_url, record = NULL)
```

Arguments

source_url	Character. Download URL (unused beyond selecting this probe).
record	Character. The dataset DOI.

Value

Named list with id and version (DataCite's version), date, and url; NULL if the record has no version.

check_dataone_version *Check a DataONE object for a newer version*

Description

DataONE system metadata links each object to the one that replaced it (obsoletedBy), so following the chain from the object the URL names reaches the newest version.

Usage

```
check_dataone_version(source_url, record = NULL)
```

Arguments

source_url	Character. DataONE member-node object URL (.../d1/mn/v2/object/<identifier>).
record	Unused.

Value

Named list with id (the newest identifier), version, date, pinned (the identifier the URL names), and url; NULL if the system metadata cannot be read.

check_dataverse_version
 Check a Dataverse dataset for its newest released version

Description

The URL either is a Dataverse API call naming the dataset by persistentId=doi:... or is the dataset DOI itself, in which case the installation serving it is the host DataCite resolves the DOI to.

Usage

```
check_dataverse_version(source_url, record = NULL)
```

Arguments

source_url	Character. Dataverse API URL or doi.org DOI.
record	Unused.

Value

Named list with id and version (major.minor), date, and url; NULL if the versions cannot be read.

check_dryad_version	<i>Check a Dryad dataset for the latest version</i>
---------------------	---

Description

Check a Dryad dataset for the latest version

Usage

```
check_dryad_version(source_url, record = NULL)
```

Arguments

source_url	Character. Dryad download URL containing a DOI, or the bare Dryad DOI (10.5061/dryad.<id>, optionally behind doi.org).
record	Unused; the URL names the dataset.

Value

Named list with id and version (both the newest version number), date, and url; NULL if the dataset cannot be read.

check_edl_version	<i>Check an EDI data package for a newer revision</i>
-------------------	---

Description

An EDI (pasta.lternet.edu) data URL names a package revision (<scope>/<identifier>/<revision>) and usually one data entity in it. An entity keeps its id across revisions, so an entity URL is compared by the entity's checksum in the revision it names and in the newest one: a revision that changes only metadata or other tables leaves it current. A URL naming the revision alone is compared by revision number.

Usage

```
check_edl_version(source_url, record = NULL)
```

Arguments

source_url	Character. EDI package data URL.
record	Unused.

Details

PASTA has required an API key for every call since 2026-07-30, read here from EDI_API_KEY as [download_edl_file\(\)](#) reads it.

Value

Named list with `id` and `pinned` (entity checksums, or revision numbers), `version` (the newest revision), `date` (its publication date), and `url`; `NULL` without a key or when PASTA cannot be read.

check_enrichment_source_version

Check one manifest enrichment entry against its upstream source

Description

Freshness is decided by comparing the upstream identity the build read against the one upstream carries now. Where the source URL pins a version, that pinned identity is what the build read, whatever was recorded. Where it floats, the build's identity comes from `upstream_id`, written into the entry when the `.vtr` was built.

Usage

```
check_enrichment_source_version(entry, probe = probe_upstream_identity)
```

Arguments

<code>entry</code>	List. Manifest enrichment entry with <code>source_url</code> , <code>source_version</code> , <code>upstream_id</code> , <code>latest</code> , <code>static</code> , and optionally <code>source_record</code> / <code>source_doi</code> .
<code>probe</code>	Function taking a URL and a host record and returning an upstream identity. Defaults to probe_upstream_identity() .

Details

A floating source with no recorded identity is still answerable when the newest upstream version is older than the build: the entry's release tag (`latest`, `YYYY.MM`) bounds the build date from below, and a build reads the newest version there is, so the build read that version.

Where none of these holds the answer is `NA`, not a guess: the recorded `source_version` is this package's own release string and the upstream version is whatever the host counts in, so comparing the two answers a different question than the one asked.

Value

Named list with `source_version`, `built_id`, `upstream_id`, `upstream_version`, `outdated`, `check_url`, and optionally `note`.

```
check_figshare_version
```

Check a Figshare article for the latest version

Description

An article URL is compared by the article's version number. A file download URL (`ndownloader.figshare.com/files/<id>`) names one immutable file, so it is compared by content: the file's md5 is the pinned identity, and the newest version's file of the same name is the upstream one (missing: `<name>` when the newest version dropped it).

Usage

```
check_figshare_version(source_url, record = NULL)
```

Arguments

<code>source_url</code>	Character. Figshare article or file download URL.
<code>record</code>	Character or NULL. The source's host record, as above.

Details

Nothing in a file URL names its article, so the article comes from the source's host record: a Figshare DOI (`10.6084/m9.figshare.<id>`), a collection DOI (`10.6084/m9.figshare.c.<id>`), whose articles are searched for the file), or a Figshare article URL.

Value

Named list with `id`, `version`, `date`, `pinned` (file URLs only), and `url`; NULL if the article cannot be read.

```
check_gbif_api_version
```

Check the GBIF backbone API for the latest update date

Description

Check the GBIF backbone API for the latest update date

Usage

```
check_gbif_api_version(source_url, record = NULL)
```

Arguments

source_url	Character. (Unused; the GBIF backbone dataset ID is fixed.)
record	Unused.

Value

Named list with id (the dataset's modified timestamp), version (that timestamp as YYYY.MM), date, and url; NULL if the dataset cannot be read.

check_gift_version	<i>Check the GIFT database for its newest version</i>
--------------------	---

Description

The GIFT API lists every database version; each description opens with the date the version was released.

Usage

```
check_gift_version(source_url, record = NULL)
```

Arguments

source_url	Character. GIFT host URL.
record	Unused.

Value

Named list with id and version (the newest version), date, and url; NULL if the versions cannot be read.

check_github_version	<i>Check a file on GitHub for its latest change</i>
----------------------	---

Description

A raw.githubusercontent.com URL serves the file at the head of a branch, so it pins nothing. The identity is the newest commit touching that path.

Usage

```
check_github_version(source_url, record = NULL)
```

Arguments

source_url	Character. raw.githubusercontent.com/<owner>/<repo>/<branch>/<path> URL.
record	Unused; the URL names the file.

Value

Named list with id (the commit SHA), version (its short form), date, and url; NULL if the commit cannot be read.

check_ipt_version	<i>Check an IPT resource for its newest published version</i>
-------------------	---

Description

An IPT archive URL names its version (v=); the same URL without it serves the newest, under a file name carrying that version.

Usage

```
check_ipt_version(source_url, record = NULL)
```

Arguments

source_url	Character. IPT archive.do?r=<resource>&v=<version> URL.
record	Unused.

Value

Named list with id and version (the newest version), date, pinned (the version the URL names), and url; NULL if the archive cannot be read.

check_kew_cvalues_version	<i>Check the Kew Plant DNA C-values database for its newest release</i>
---------------------------	---

Description

Check the Kew Plant DNA C-values database for its newest release

Usage

```
check_kew_cvalues_version(source_url, record = NULL)
```

Arguments

source_url	Character. A <code>cvalues.science.kew.org</code> URL.
record	Unused.

Value

Named list with `id` and `version` (the release number), `date` (the release month), and `url`; `NULL` if the release history cannot be read.

check_last_modified	<i>Check a file for its Last-Modified date</i>
---------------------	--

Description

For a host that serves a static file and says when it last changed. The header stamp is kept whole as the identity and shortened to `YYYY.MM` only for display: two files a fortnight apart share a month, and an identity that cannot tell them apart reports a moved file as unchanged.

Usage

```
check_last_modified(source_url, record = NULL)
```

Arguments

source_url	Character. File URL.
record	Unused.

Value

Named list with `id` (the Last-Modified stamp), `version` (that stamp as `YYYY.MM`), `date`, and `url`; `NULL` if the header is absent.

check_lcvp_version	<i>Check the LCVp data package for the latest version</i>
--------------------	---

Description

LCVP is distributed as an R data package; its data version is the `Version` field of the package `DESCRIPTION` on GitHub.

Usage

```
check_lcvp_version(source_url)
```

Arguments

source_url Character. LCVP tab_lcvp.rda download URL (unused; the DESCRIPTION location is derived from the fixed repository).

Value

Named list with version and url.

check_ncbi_archive_version

Check the NCBI Taxonomy monthly archive for a newer dump

Description

NCBI rewrites its rolling new_taxdump daily, and keeps one dump a month in taxdump_archive/ as new_taxdump_<YYYY-MM-DD>.zip. A source pinned to an archived dump compares its date against the newest one there.

Usage

```
check_ncbi_archive_version(source_url, record = NULL)
```

Arguments

source_url Character. Archive URL naming its dump date.
record Unused.

Value

Named list with id, version and date (the newest dump's date), pinned (the date the URL names), and url; NULL if the archive listing cannot be read.

check_ott_version

Check the Open Tree Taxonomy for its newest release

Description

Check the Open Tree Taxonomy for its newest release

Usage

```
check_ott_version(source_url, record = NULL)
```

Arguments

source_url	Character. OTT archive URL naming its release (.../ott/ott<version>/...).
record	Unused.

Value

Named list with id and version (the newest release), pinned (the release the URL names), and url; NULL if the listing cannot be read.

check_pangaea_version *Check a PANGAEA dataset for changes and successors*

Description

A PANGAEA DOI names one dataset, which is replaced by a new DOI rather than edited; the meta-data record states the data's last modification, its status, and any DataCite relation to a successor.

Usage

```
check_pangaea_version(source_url, record = NULL)
```

Arguments

source_url	Character. doi.pangaea.de URL of the dataset.
record	Unused.

Value

Named list with id (last modification, status and any successor), version, date, and url; NULL if the record cannot be read.

check_phaidra_version *Check a Phaidra object for a newer version*

Description

Phaidra (University of Vienna) objects are immutable; a new version is a new object listed under the old one's versions. The portal sits behind an Anubis proof-of-work page, which .anubis_fetch() clears.

Usage

```
check_phaidra_version(source_url, record = NULL)
```

Arguments

source_url	Character. Phaidra object URL (.../object/o:<n>/...).
record	Unused.

Value

Named list with id (the newest object), version, date, pinned (the object the URL names), and url; NULL if the object cannot be read.

check_rfishbase_version

Check an rfishbase server for its newest release

Description

rfishbase reads FishBase and SeaLifeBase snapshots published as release folders of one Hugging Face dataset, and takes the newest by default.

Usage

```
check_rfishbase_version(source_url, record = NULL)
```

Arguments

source_url	Character. fishbase.ropensci.org or sealifebase.ropensci.org.
record	Unused.

Value

Named list with id and version (the release label), date (the dataset's last change, an upper bound on the release date), and url; NULL if the releases cannot be read.

check_seanoe_version *Check a SEANOE dataset for its file set*

Description

SEANOE updates a dataset in place under one DOI, adding the new files beside the old ones. The set of file names is therefore the dataset's identity: a new version shows up as a file the set did not have.

Usage

```
check_seanoe_version(source_url, record = NULL)
```

Arguments

source_url	Character. SEANOE dataset URL (.../data/<n>/<id>/).
record	Unused.

Value

Named list with id (the sorted file names), version (the file count), and url; NULL if the dataset cannot be read.

check_wayback_version *Check a Wayback Machine snapshot for a newer capture*

Description

A snapshot URL (web.archive.org/web/<timestamp>id_/<original>) fixes one capture of a page that may since have disappeared. Each capture carries a digest of the bytes it holds, so the capture the URL names and the newest capture of the same original compare by content: a re-capture of an unchanged file is not a new version.

Usage

```
check_wayback_version(source_url, record = NULL)
```

Arguments

source_url	Character. Wayback snapshot URL.
record	Unused.

Value

Named list with id (the newest capture's digest), version (its timestamp), date, pinned (the digest of the capture the URL names), and url; NULL if the capture index cannot be read.

check_wst_version *Check the World Spider Trait database for new datasets*

Description

The database grows by datasets, each uploaded with a date and a record count, so their number, their records and the newest upload together say whether the export would read differently.

Usage

```
check_wst_version(source_url, record = NULL)
```

Arguments

source_url	Character. A spidertraits.sci.muni.cz URL.
record	Unused.

Value

Named list with id, version, date (the newest upload), and url; NULL if the dataset list cannot be read.

check_zenodo_version	<i>Check a Zenodo record for the latest version</i>
----------------------	---

Description

A Zenodo record is immutable, so querying the pinned record reports the day that record was published however many newer versions its concept has since gained. The versions/latest endpoint resolves the pinned record to the newest one in the same concept. A URL naming the concept record itself floats: Zenodo serves its newest version, so it pins nothing.

Usage

```
check_zenodo_version(source_url, record = NULL)
```

Arguments

source_url	Character. Zenodo record or file download URL.
record	Unused; the URL names the record.

Details

Where the URL names one file of the record, the identity is that file's checksum, read from the pinned and the newest record alike: a new version that only adds a metadata file leaves the data the build reads unchanged. A file the newest record no longer carries reads as missing: <key>.

Value

Named list with id, version, date, pinned (absent for a concept record), and url; NULL if the record cannot be read.

colxr_col_publications

Publication references of the COL Base Release, keyed for COL XR

Description

COL XR keeps the Base Release's identifiers, so a usage's publication reference can be read from the COL build by `taxon_id`. Only rows carrying a reference are kept.

Usage

```
colxr_col_publications(col_path)
```

Arguments

`col_path` Path to a COL .vtr carrying `name_published_in`.

Value

A data.frame of `taxon_id`, `canonical_name`, `authorship`, `name_published_in` and `year`.

colxr_gbif_crosswalk *Legacy GBIF keys for each COL XR usage, from the GBIF backbone*

Description

COL XR replaced the taxonomy behind GBIF.org, but GBIF serves occurrence records only for the numeric keys of its legacy backbone. This reads the gbif backbone and groups its usages by canonical name, rank and authorship (see `crosswalk_authorship()`). GBIF files the records of a synonym under its accepted taxon, so a synonym usage contributes its accepted taxon's key. A name several GBIF usages share (a homonym, or one name GBIF split itself) maps to the set of their keys, `|`-delimited in ascending order.

Usage

```
colxr_gbif_crosswalk(gbif_path)
```

Arguments

`gbif_path` Character. Path to the gbif .vtr.

Value

A `colxr_gbif_crosswalk`: the key sets grouped by full authorship and by outer authorship, for [colxr_gbif_lookup\(\)](#).

colxr_gbif_lookup	<i>Look COL XR usages up in a GBIF crosswalk</i>
-------------------	--

Description

In order: the usage's full authorship against GBIF's, its outer authorship against GBIF's outer authorship, and, where GBIF records no authorship for the name, the name and rank alone. A GBIF usage that names a different author is a different name and is never matched.

Usage

```
colxr_gbif_lookup(crosswalk, canonical_name, authorship, taxon_rank)
```

Arguments

crosswalk A `colxr_gbif_crosswalk()`.
canonical_name, authorship, taxon_rank
 Character vectors of the COL XR usages.

Value

Character vector of | -delimited GBIF keys, NA where none.

colxr_latest_release	<i>Resolve the latest Catalogue of Life Extended Release</i>
----------------------	--

Description

Queries ChecklistBank for the most recently issued COL XR dataset whose Darwin Core export can be downloaded. Each monthly release carries its own dataset key, so the key is looked up rather than hard-coded. ChecklistBank generates an export only when a logged-in account asks for one, so when the newest release has none, one is requested with `colxr_request_export()`. When that cannot run (no GBIF_USER / GBIF_PWD) or does not finish, the newest release with an export is built instead.

Usage

```
colxr_latest_release(
  verbose = TRUE,
  export_ready = colxr_export_ready,
  request_export = function(key) {
    colxr_request_export(key, verbose = verbose)
  }
)
```

Arguments

verbose	Logical.
export_ready	Function of a dataset key returning TRUE when its export can be downloaded.
request_export	Function of a dataset key that requests its export and returns TRUE once it can be downloaded.

Value

A list with key, alias, version and issued.

colxr_request_export	<i>Request a ChecklistBank export and wait for it</i>
----------------------	---

Description

ChecklistBank generates a dataset's export only when a logged-in GBIF account asks for one, so a release can be issued and never become downloadable. This starts a DwCA export job with the account in GBIF_USER / GBIF_PWD and polls it until it finishes.

Usage

```
colxr_request_export(key, poll = 30, timeout = 3600, verbose = TRUE)
```

Arguments

key	Character. ChecklistBank dataset key.
poll	Seconds between status checks.
timeout	Seconds before giving up.
verbose	Logical.

Value

TRUE when the export finished, FALSE when no account is configured or the job did not finish.

count_gbif_occurrences	<i>Count GBIF occurrence records per taxon key</i>
------------------------	--

Description

Count GBIF occurrence records per taxon key

Usage

```
count_gbif_occurrences(  
  keys,  
  batch_size = 200L,  
  facet_limit = 50000L,  
  verbose = TRUE  
)
```

Arguments

keys	Character vector of GBIF taxon keys.
batch_size	Integer. Keys per request.
facet_limit	Integer. Facet entries per request; a batch whose facet comes back full is re-counted key by key.
verbose	Logical.

Value

A data.frame with taxon_id and n_occurrences.

count_vtr_rows	<i>Count rows in a .vtr file</i>
----------------	----------------------------------

Description

Count rows in a .vtr file

Usage

```
count_vtr_rows(vtr_path)
```

Arguments

vtr_path	Character.
----------	------------

Value

Integer.

create_delta	<i>Create a binary diff between two .vtr files</i>
--------------	--

Description

Besides the patch, writes <delta_path>.base: one line holding the content id (md5) of old_path. A patch applies only to the exact bytes it was cut against, and a release tag does not identify those bytes (a re-cut reuses it), so the manifest records this id as delta_from_content_id and the run-time patches only a local build carrying it.

Usage

```
create_delta(old_path, new_path, delta_path)
```

Arguments

old_path	Character. Path to the previous-version .vtr.
new_path	Character. Path to the new-version .vtr.
delta_path	Character. Output path for the .xdelta file.

Value

The delta path (invisibly), or NULL if xdelta3 is unavailable.

delim_chunk_feed	<i>Feed a delimited file to build_vtr_streamed() a chunk at a time</i>
------------------	--

Description

Reads a header once, then hands back successive blocks of rows, so a file larger than memory can be normalized and staged block by block.

Usage

```
delim_chunk_feed(
  path,
  normalize,
  chunk_rows = 500000L,
  sep = "\t",
  quote = "",
  encoding = "UTF-8",
  select = NULL,
  na_strings = c("", "NA"),
  col_names = NULL,
  file_encoding = NULL,
  verbose = TRUE
)
```

Arguments

path	Character. Path to the delimited file.
normalize	A function taking one raw chunk (a <code>data.frame</code> with the file's own column names) and returning it in the normalized schema.
chunk_rows	Integer. Rows to read per chunk.
sep	Character. Field separator.
quote	Character. Quoting character, or "" to disable quoting.
encoding	Character. Passed to <code>data.table::fread()</code> .
select	Character vector of columns to read, or <code>NULL</code> for all.
na_strings	Character vector read as NA.
col_names	Character vector naming the columns of a file that carries no header row, in file order, or <code>NULL</code> when the first line is a header.
file_encoding	Character or <code>NULL</code> . The encoding of the file's bytes: character columns are decoded from it to UTF-8 once the block is parsed, and a byte that does not form a character in it is dropped. WFO declares "UTF-8", and its <code>taxonRemarks</code> is cut at a fixed byte width that leaves the lead byte of a split character at the end of 8 fields. A file needing this is read block by block whether or not it is quoted, since the decoding happens there.
verbose	Logical.

Details

quote, encoding and file_encoding must be set to whatever the backbone's whole-file reader uses. They decide how the text is parsed, not just how fast it is read: WoRMS ships its TSV with every field wrapped in double quotes, so reading it with quoting disabled leaves the quote characters inside the names and breaks every downstream join, while a source carrying genuine embedded quotes in informal names needs quoting disabled to keep them.

quote also decides how the file is cut. With quoting disabled a newline always ends a record, so blocks are taken by row offset and `data.table::fread()` reads each directly, which is the faster path and the one the unquoted sources use.

With quoting enabled that stops holding: a field may contain a newline, and then a line offset is not a record offset. WoRMS carries 4,626 newlines inside quoted `namePublishedIn` fields, and every reader's row argument counts lines – `fread()`'s `skip` does, and so does `read.delim(nrows=)` through `scan()`'s `nlines` – so a line-offset read is 445 records adrift by record 200,000 and silently repeats rows. A quoted source is therefore cut only where the running count of quote characters is even, which is the one place a record can end, and the number of records that implies is checked against the number the parser returns.

Lines are taken at the byte level, split on LF alone. R's own line reader recognizes a lone CR as a terminator as well, and WoRMS holds 59 of them inside fields: read as text it manufactures 57 lines the file does not have and drops 49 quote characters, which flips the parity and is why `utils::read.delim()` gives up on that file partway through.

A field may also contain "", which RFC 4180 defines as one literal quote. `utils::read.delim()` collapses it and `fread()` returns it doubled (`Rdatatable/data.table#1109`); WFO has 4,117 such fields, and leaving them doubled would put stray quote characters in the published values, which is

the defect worms-2026.07 was cut to fix. Rather than let the choice of reader decide that, the blocks are parsed with `fread()` and the unescaping is done here, so it holds whichever source is being read.

Value

A function of no arguments suitable as the feed of `build_vtr_streamed()`.

delim_fk_lookup	<i>Build an id -> value lookup for the ids a file's foreign keys reference</i>
-----------------	---

Description

A backbone that denormalizes its higher classification stores foreign keys rather than names, so normalizing a row needs a name that lives in some other row. GBIF is the case: kingdom, phylum, class, order and family all arrive as `*_key` columns pointing at rows that the build then drops.

Usage

```
delim_fk_lookup(
  path,
  id_col,
  value_col,
  key_cols,
  chunk_rows = 1000000L,
  sep = "\t",
  quote = "",
  encoding = "UTF-8",
  na_strings = c("", "NA"),
  col_names = NULL,
  verbose = TRUE
)
```

Arguments

<code>path</code>	Character. Path to the delimited file.
<code>id_col</code>	Character. Column holding the row's own identifier.
<code>value_col</code>	Character. Column whose value the keys resolve to.
<code>key_cols</code>	Character vector of foreign-key columns.
<code>chunk_rows</code>	Integer. Rows to read per scan chunk.
<code>sep, quote, encoding, na_strings, col_names</code>	As in <code>delim_chunk_feed()</code> .
<code>verbose</code>	Logical.

Details

The whole id -> name map would be as large as the backbone, but only the ids actually referenced are ever looked up, and those are the few thousand rows at the ranks the keys point to. Two scans find them without holding the map: the first collects the distinct keys, the second keeps the id/value pair of each row a key names. What comes back is small enough to sit beside a chunk, which makes the rest of the normalization row-local.

The result is exactly the lookup a whole-table self-join would produce; it is restricted by which ids are referenced, never by an assumption about the rank a key points at.

Value

A named character vector of value, named by id.

download_algaebase	<i>Fetch all AlgaeBase records via /nameusage/search</i>
--------------------	--

Description

Fetch all AlgaeBase records via /nameusage/search

Usage

```
download_algaebase(verbose = TRUE)
```

Arguments

verbose Logical.

Value

A list of record objects (raw JSON list-of-lists).

download_and_unzip	<i>Download a ZIP and extract, returning the path to a matching file</i>
--------------------	--

Description

Downloads url to dest_dir/source.zip (if not cached), extracts into dest_dir/extracted/, and returns the first file matching pattern. If pattern is NULL, returns the extraction directory itself.

Usage

```
download_and_unzip(url, dest_dir, pattern = NULL)
```

Arguments

url	Character. URL to a ZIP archive.
dest_dir	Character. Directory for download and extraction.
pattern	Character or NULL. Regex to match the target file inside the ZIP. If NULL, returns the extraction directory itself.

Value

Path to the matched file, or the extraction directory if pattern is NULL.

download_avilist	<i>Download the AviList extended checklist workbook</i>
------------------	---

Description

Download the AviList extended checklist workbook

Usage

```
download_avilist(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the downloaded .xlsx.

download_cf_file	<i>Download a Cloudflare-fronted file via the curl_cffi helper</i>
------------------	--

Description

Downloads url into dest_dir/filename with a browser TLS impersonation that passes Cloudflare's "Just a moment" challenge. A host running a JavaScript challenge, which no impersonation can satisfy, is instead cleared once in a visible Chrome window and the resulting cookies reused for 12h; that rung needs the Python package nodriver. Cached: skips if the destination exists and is non-empty.

Usage

```
download_cf_file(url, dest_dir, filename)
```

Arguments

url	Character. URL to download.
dest_dir	Character. Directory to save into. Created if missing.
filename	Character. Output filename.

Value

Path to the downloaded file.

download_col	<i>Download and extract the COL DwC-A</i>
--------------	---

Description

Download and extract the COL DwC-A

Usage

```
download_col(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the extracted COL directory.

download_colxr	<i>Download and extract the COL XR Darwin Core Archive</i>
----------------	--

Description

Download and extract the COL XR Darwin Core Archive

Usage

```
download_colxr(dest = tempdir(), key = NULL, verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
key	Character or NULL. ChecklistBank dataset key. Resolved from colxr_latest_release() when NULL.
verbose	Logical.

Value

Path to the extracted TSV.

download_curl_file	<i>Download a file via curl</i>
--------------------	---------------------------------

Description

Downloads url into dest_dir/filename if the destination does not already exist (or is empty). Follows redirects, sets a User-Agent header, and optionally a Referer.

Usage

```
download_curl_file(  
  url,  
  dest_dir,  
  filename,  
  referer = NULL,  
  user_agent = NULL,  
  extra_headers = NULL,  
  max_tries = 4L  
)
```

Arguments

url	Character. URL to download.
dest_dir	Character. Directory to save into. Created if missing.
filename	Character. Output filename.
referer	Character or NULL. Optional Referer header.
user_agent	Character or NULL. Override the default User-Agent.
extra_headers	Named character vector or NULL. Additional request headers (e.g. a Cookie for a session-authenticated source).
max_tries	Integer. Download attempts before giving up. Large live exports (e.g. the World Spider Trait database) intermittently drop the TLS connection mid-transfer; each retry uses exponential backoff.

Value

Path to the downloaded file.

download_dryad_file	<i>Download a Dryad data file (handles the Anubis bot challenge)</i>
---------------------	--

Description

Resolves the newest version of a Dryad dataset DOI to a data file matching `file_pattern`, then downloads it through `.anubis_fetch()`. Cached: skips the download if `dest_dir/filename` already exists and is non-empty.

Usage

```
download_dryad_file(doi, dest_dir, filename, file_pattern = "\\*.csv$")
```

Arguments

<code>doi</code>	Character. Dryad dataset DOI (e.g. "10.5061/dryad.1cv08").
<code>dest_dir</code>	Character. Directory to save into. Created if missing.
<code>filename</code>	Character. Output filename.
<code>file_pattern</code>	Character. Regex to pick the file from the version's file list (matched against the file path, case-insensitive).

Value

Path to the downloaded file.

download_edi_file	<i>Download a file from EDI's PASTA repository, authenticated</i>
-------------------	---

Description

EDI has required authentication for all REST API access since 2026-07-30 (an anti-DoS/scraping measure; see edirepository.org/news/news-20260727.00). Anonymous requests to `pasta.lternet.edu` now get a 403 ("Public Access is not authorized"). This wraps `download_curl_file()` with EDI's documented auth: an API Access Key appended as a `?key=` query parameter (the supported mechanism for "coded workflows, automated scripts, and other integrations", per <https://edirepository.org/resources/iam>), read from the `EDI_API_KEY` env var. `EDI_EDI_TOKEN` (a raw edi-token session JWT, from a logged-in browser) is an interactive fallback – it expires in days, so it is not suitable for a scheduled/CI rebuild.

Usage

```
download_edi_file(url, dest_dir, filename, max_tries = 4L)
```

Arguments

url	Character. URL to download.
dest_dir	Character. Directory to save into. Created if missing.
filename	Character. Output filename.
max_tries	Integer. Download attempts before giving up. Large live exports (e.g. the World Spider Trait database) intermittently drop the TLS connection mid-transfer; each retry uses exponential backoff.

Value

Path to the downloaded file.

download_euromed	<i>Download the frozen Euro+Med CDM snapshot</i>
------------------	--

Description

Fetches the crawler's NDJSON snapshot and node-path table from the release. If a local crawl directory already holds both files (the machine that ran the crawl), they are used directly.

Usage

```
download_euromed(dest = tempdir(), local_dir = NULL, verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
local_dir	Character or NULL. A local crawl output directory to use instead of downloading (default ~/dev/taxify-crawls/euromed).
verbose	Logical.

Value

Named list with jsonl and nodes paths.

download_evc	<i>Download the frozen EuroVegChecklist snapshot</i>
--------------	--

Description

Fetches the unit list and the harvested synonymy from the evc-snapshot-<version> release. A local crawl directory holding both files (the machine that ran the crawl) is used directly.

Usage

```
download_evc(dest = tempdir(), local_dir = NULL, verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
local_dir	Character or NULL. A local crawl output directory to use instead of downloading (default ~/dev/taxify-crawls/evc).
verbose	Logical.

Value

Named list with synonyms, units and synonyms_2016 paths.

download_fungorum	<i>Download and extract the Species Fungorum Plus ColDP archive</i>
-------------------	---

Description

Download and extract the Species Fungorum Plus ColDP archive

Usage

```
download_fungorum(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the directory containing the extracted ColDP files.

download_gbif	<i>Download the GBIF backbone simple.txt.gz</i>
---------------	---

Description

Download the GBIF backbone simple.txt.gz

Usage

```
download_gbif(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the .gz file.

download_gbif_api_pages	<i>Fetch paginated GBIF API results</i>
-------------------------	---

Description

Pages through a GBIF API endpoint that uses offset + limit and an endOfRecords flag. Returns a combined data.frame of \$results payloads.

Usage

```
download_gbif_api_pages(base_url, params, limit = 1000L, max_pages = 100L)
```

Arguments

base_url	Character. GBIF API endpoint.
params	Named list. Query parameters (excluding offset/limit).
limit	Integer. Page size.
max_pages	Integer. Maximum pages to fetch.

Value

A data.frame of combined results. Empty data.frame if the endpoint returned no rows.

download_gbif_occurrence_counts	<i>Download the published GBIF occurrence-count snapshot</i>
---------------------------------	--

Description

Download the published GBIF occurrence-count snapshot

Usage

```
download_gbif_occurrence_counts(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the downloaded .tsv.gz.

download_groot	<i>Download both GRoot files (species aggregate + full per-record version)</i>
----------------	--

Description

The aggregate drives every trait; the full version is needed to repair specific_root_area (see [parse_groot\(\)](#)). The full download is best-effort: if it fails, the aggregate SRA is used as-is.

Usage

```
download_groot(url, dest)
```

Arguments

url	Character. The aggregate ZIP URL (from the registry).
dest	Character. Download directory.

Value

Named character vector c(aggregate = <csv>, full = <csv or NA>).

download_itis	<i>Download and extract the ITIS SQLite database</i>
---------------	--

Description

Download and extract the ITIS SQLite database

Usage

```
download_itis(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the extracted SQLite database file.

download_lcvp	<i>Download the LCVP tab_lcvp data file</i>
---------------	---

Description

Download the LCVP tab_lcvp data file

Usage

```
download_lcvp(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the downloaded tab_lcvp.rda.

download_lpsn	<i>Download the LPSN ColDP archive from ChecklistBank</i>
---------------	---

Description

Download the LPSN ColDP archive from ChecklistBank

Usage

```
download_lpsn(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the directory holding the extracted NameUsage.tsv.

download_mdd	<i>Download and unpack the Mammal Diversity Database archive</i>
--------------	--

Description

Download and unpack the Mammal Diversity Database archive

Usage

```
download_mdd(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the directory holding the extracted CSVs.

download_meow*Download the MEOW ecoregion GeoJSON snapshot*

Description

Download the MEOW ecoregion GeoJSON snapshot

Usage

```
download_meow(dest = tempfile(fileext = ".geojson"), verbose = TRUE)
```

Arguments

dest	Character. Output path for the downloaded GeoJSON.
verbose	Logical.

Value

The path to the downloaded GeoJSON (invisibly).

download_ncbi*Download and extract NCBI taxdump*

Description

Download and extract NCBI taxdump

Usage

```
download_ncbi(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the extraction directory.

download_ott	<i>Download and extract OTT taxonomy</i>
--------------	--

Description

Download and extract OTT taxonomy

Usage

```
download_ott(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the extraction directory containing taxonomy.tsv.

download_reptiledb	<i>Download the Reptile Database bulk files</i>
--------------------	---

Description

Download the Reptile Database bulk files

Usage

```
download_reptiledb(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Named list with paths: taxa, synonyms, checklist.

download_vegcz	<i>Download the frozen Pladias vegetation snapshot</i>
----------------	--

Description

Download the frozen Pladias vegetation snapshot

Usage

```
download_vegcz(dest = tempdir(), local_dir = NULL, verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
local_dir	Character or NULL. A local crawl output directory to use instead of downloading (default ~/dev/taxify-crawls/vegcz).
verbose	Logical.

Value

Named list with synonyms and units paths.

download_wcvp	<i>Download and extract the WCVp names file</i>
---------------	---

Description

Download and extract the WCVp names file

Usage

```
download_wcvp(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the extracted wcvp_names.csv.

download_wfo	<i>Download and extract the WFO classification file</i>
--------------	---

Description

Download and extract the WFO classification file

Usage

```
download_wfo(dest = tempdir(), verbose = TRUE, url = NULL)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.
url	Character or NULL. Backbone archive URL; the newest edition's, from wfo_latest_edition() , when NULL.

Value

Path to the extracted classification file.

download_wgsrpd	<i>Download the WGSRPD Level 3 GeoJSON</i>
-----------------	--

Description

Download the WGSRPD Level 3 GeoJSON

Usage

```
download_wgsrpd(dest = tempfile(fileext = ".geojson"), verbose = TRUE)
```

Arguments

dest	Character. Output path for the downloaded GeoJSON.
verbose	Logical.

Value

The path to the downloaded GeoJSON (invisibly).

download_wiley_supplement

Download a Wiley/Atypon supporting-information file

Description

Wiley's supplement endpoint (/action/downloadSupplement) returns 403 to a cold request: it requires session cookies set by first loading the article page and an article Referer. The bundled cf_fetch.py wiley mode primes the session and sends the correct headers through a browser TLS impersonation, clearing both the Cloudflare tier and the cookie/Referer check. Cached: skips if the destination exists and is non-empty.

Usage

```
download_wiley_supplement(article_url, doi, sup_file, dest_dir, filename)
```

Arguments

article_url	Character. The article landing-page URL (used to prime the session and as Referer), e.g. a .../doi/10.1002/ecy.1745 URL.
doi	Character. The article DOI.
sup_file	Character. The supplement file name, e.g. ecy1745-sup-0001-DataS1.zip.
dest_dir	Character. Directory to save into. Created if missing.
filename	Character. Output filename.

Value

Path to the downloaded file.

download_worms

Download and extract WoRMS DwC-A

Description

Download and extract WoRMS DwC-A

Usage

```
download_worms(dest = tempdir(), verbose = TRUE)
```

Arguments

dest	Character. Destination directory.
verbose	Logical.

Value

Path to the extracted DwC-A directory.

enrichment_emergency_fallback

Build enrichment from source and return the raw data.frame

Description

Same as `build_enrichment()` but returns the parsed `data.frame` instead of writing a `.vtr` file. Useful when the runtime needs the raw data in memory (e.g. emergency fallback wired into `taxify`).

Usage

```
enrichment_emergency_fallback(name, verbose = TRUE)
```

Arguments

name	Character. Enrichment identifier.
verbose	Logical. Default TRUE.

Value

`data.frame` with `canonical_name` + trait columns.

enrichment_trait_cols *Trait columns of an enrichment*

Description

The columns `taxify`'s runtime exposes from an enrichment `.vtr`: every column except the join keys (`canonical_name`, `accepted_name`, `genus`) and the group column, which is surfaced through `available_groups` and the door's group argument rather than as a trait. This is what a manifest entry's `trait_cols` records.

Usage

```
enrichment_trait_cols(columns, group_col = NULL)
```

Arguments

columns	Character vector of <code>.vtr</code> column names.
group_col	Character or NULL. The enrichment's group column.

Value

Character vector of trait column names, in `.vtr` order.

gbif_count_keys	<i>Taxon keys whose occurrence count can decide a GBIF pick</i>
-----------------	---

Description

Taxon keys whose occurrence count can decide a GBIF pick

Usage

```
gbif_count_keys(vtr_path)
```

Arguments

vtr_path Character. Path to a built GBIF backbone .vtr.

Value

Character vector of taxon keys: every record sharing a matching key with a record of another accepted taxon, plus the accepted targets of those records.

gen_spe	<i>Six-letter GEN_SPE species code</i>
---------	--

Description

The BETSI-derived code-keyed matrices (INRAE UU2FQT / UCYSLH, Bonfanti 2022) encode a species as the first three letters of the genus and the first three of the specific epithet, uppercased and joined by an underscore: "Brachystomella parvula" becomes "BRA_PAR". Hyphens in the epithet are dropped before the first three letters are taken. Returns NA for anything that is not a capitalised Genus species binomial.

Usage

```
gen_spe(binomial)
```

Arguments

binomial Character vector of "Genus species" names.

Value

Character vector of GEN_SPE codes, NA where the input is not a capitalised Genus species binomial.

Examples

```
gen_spe(c("Brachystomella parvula", "Allacma gallica", "Onychiurus"))
```

gunzip_file	<i>Decompress a gzipped file to a plain file</i>
-------------	--

Description

A chunked reader seeks by row offset, and every seek into a .gz restarts the decompression, so a gzipped source is expanded once up front instead.

Usage

```
gunzip_file(gz_path, out_path, verbose = TRUE)
```

Arguments

gz_path	Character. Path to the .gz file.
out_path	Character. Path to write the decompressed file to.
verbose	Logical.

Value

out_path, invisibly.

harvest_ckan_datastore	<i>Harvest an entire CKAN datastore resource past the offset window</i>
------------------------	---

Description

CKAN's elasticsearch-backed datastore caps offset at max_result_window (10000); the helper pages the full table via the after (search_after) cursor, through Cloudflare, writing one JSON record per line. Cached.

Usage

```
harvest_ckan_datastore(api_base, resource_id, dest_dir, filename)
```

Arguments

api_base	Character. CKAN action root, e.g. https://data.nhm.ac.uk/api/3/action.
resource_id	Character. Datastore resource id.
dest_dir	Character. Directory to save into. Created if missing.
filename	Character. Output filename (a .jsonl).

Value

Path to the downloaded JSONL file.

harvest_plazi_dwca	<i>Harvest the Plazi Darwin Core archives covering a taxon</i>
--------------------	--

Description

Plazi republishes each marked-up paper as its own GBIF checklist dataset, so a taxon's treatments are spread over hundreds of small archives rather than one bulk file. Discovery is the union of two GBIF dataset-search filters because neither is complete alone: taxonKey depends on GBIF having indexed the dataset's taxonomic coverage, and the free-text query depends on the paper's title or abstract naming the taxon. For class Collembola the two return 210 and 489 datasets and their union is 508.

Usage

```
harvest_plazi_dwca(taxon_key, dest_dir, query = NULL, max_datasets = 20000L)
```

Arguments

taxon_key	Integer. GBIF backbone key of the taxon (10713444 is class Collembola).
dest_dir	Character. Directory to cache the archives in. Created if missing.
query	Character or NULL. Free-text term for the second discovery pass, usually the taxon name.
max_datasets	Integer. Safety bound on the number of datasets.

Details

Each dataset is resolved to its DWC_ARCHIVE endpoint and cached as <dataset key>.zip, so a re-run only fetches what is new.

Value

Path to the directory holding the cached archives.

has_xdelta3	<i>Check if xdelta3 is available on PATH</i>
-------------	--

Description

Check if xdelta3 is available on PATH

Usage

```
has_xdelta3()
```

Value

Logical.

inrae_genus_dict	<i>INRAE genus-abbreviation dictionary</i>
------------------	--

Description

The verified TOKEN -> genus mapping for the bespoke genus abbreviations in the INRAE UU2FQT / UCYSLH code-keyed matrices, read from `inst/extdata/betsi/inrae_genus_abbr.csv`. Each token is confirmed by a named Collembola species in COLTRAIT (Salmon et al. 2014) or the built reference pool; unconfirmable tokens are omitted so `resolve_betsi_codes()` leaves them unresolved. Suitable to pass as the `genus_dict` argument.

Usage

```
inrae_genus_dict()
```

Value

Named character vector mapping token to genus.

See Also

`resolve_betsi_codes()`

list_backends	<i>List available backend builders</i>
---------------	--

Description

The taxonomic backbones only. This is the set enrichment name resolution unions over, so reference geometry (`list_geometry()`) is not part of it even though `build_backend()` builds both.

Usage

```
list_backends()
```

Value

Character vector of backend identifiers.

list_betsi_recovery	<i>List the BETSI-recovery enrichments</i>
---------------------	--

Description

The per-taxon assets rebuilt from published BETSI-derived matrices. Each is a normal entry in the enrichment build registry; this lists the recovery subset [build_betsi_recovery\(\)](#) accepts.

Usage

```
list_betsi_recovery()
```

Value

Character vector of recovery enrichment names.

See Also

[build_betsi_recovery\(\)](#), [parse_betsi_recovery\(\)](#)

list_build_only_enrichments	<i>List build-only (non-redistributable) candidate trait sources</i>
-----------------------------	--

Description

Candidate enrichment sources that taxifydb deliberately does **not** build into a .vtr, record in manifest.json, or place in the cross-source trait registry, because each source's own current licence permits citation and personal/scientific use only, not third-party redistribution. Unlike the scrape-only sources in [list_scrape_only_enrichments\(\)](#) (which taxify still reaches at runtime on the user's own machine), these have no runtime accessor. The catalog records the verified licence decision so a source that resurfaces is not re-evaluated from scratch; a source moves into the build registry only if its live licence is relicensed to permit redistribution.

Usage

```
list_build_only_enrichments()
```

Value

A data.frame with columns name, source, organism_groups, candidate_trait, source_url, license, access, reason, attribution, and issue.

See Also

[list_scrape_only_enrichments\(\)](#), [list_enrichments\(\)](#)

list_enrichments	<i>List available enrichment builders</i>
------------------	---

Description

List available enrichment builders

Usage

```
list_enrichments()
```

Value

Character vector of enrichment identifiers.

list_geometry	<i>List available reference-geometry builders</i>
---------------	---

Description

Boundary polygons taxify reads for the region= / coords= constraint. Published and versioned like a backbone, but they carry no taxonomic names.

Usage

```
list_geometry()
```

Value

Character vector of geometry identifiers.

list_scrape_only_enrichments	<i>List scrape-only (non-redistributed) trait sources</i>
------------------------------	---

Description

taxifydb builds no .vtr for these sources because they cannot be redistributed. Pignatti's values originate in a copyrighted publication and are accessed on demand by taxify's add_pignatti() through the TR8 package (which ships a copy under its own licence). This returns the catalog of those sources.

Usage

```
list_scrape_only_enrichments()
```

Value

A data.frame with columns name, tr8_db, taxify_fn, license, and reason.

normalize_backbone	<i>Normalize a raw backbone data.frame to the unified schema</i>
--------------------	--

Description

Renames source-specific columns to canonical names and ensures consistent types and formatting (uppercase ranks/statuses, trimmed whitespace, etc.).

Usage

```
normalize_backbone(df, col_map, extra_cols = NULL)
```

Arguments

- df The raw data.frame from a backend's download/parse step.
- col_map A named list mapping canonical names to source column names: taxon_id, canonical_name, taxon_rank, taxonomic_status, accepted_name_usage_id, family, genus, specific_epithet, authorship, infraspecific_epithet, and, for a source that links a combination to its basionym, original_name_usage_id (the basionym's taxon_id, read by taxify to place an unplaced combination).
- extra_cols Optional named list of additional columns to keep (canonical_name = source_name).

Value

A data.frame with standardized column names and formatting.

normalize_colxr	<i>Normalize one block of COL XR rows to the unified schema</i>
-----------------	---

Description

Split out of `read_colxr()` so the streaming build can apply it to a chunk at a time. Nothing here depends on rows outside the block.

Usage

```
normalize_colxr(df, gbif_crosswalk = NULL, publications = NULL, verbose = TRUE)
```

Arguments

df	A data.frame of raw COL XR rows, with the export's own column names.
gbif_crosswalk	A <code>colxr_gbif_crosswalk()</code> , or NULL to leave gbif_key unset.
publications	A <code>colxr_col_publications()</code> table, or NULL to carry no publication columns.
verbose	Logical.

Value

A normalized data.frame.

normalize_gbif	<i>Normalize one block of GBIF rows to the unified schema</i>
----------------	---

Description

Split out of `read_gbif()` so the streaming build can apply it to a chunk at a time. Everything here depends only on the row in hand once higher is supplied, which is the whole reason that lookup is a separate argument: the *_key columns point at rows anywhere in the file.

Usage

```
normalize_gbif(df, higher, counts = NULL, verbose = TRUE)
```

Arguments

df	A data.frame of raw GBIF rows, named by <code>.gbif_col_names</code> .
higher	Named character vector of canonical names by id, from <code>gbif_higher_lookup()</code> .
counts	Named numeric vector of occurrence counts by taxon key, from <code>read_gbif_occurrence_counts()</code> , or NULL. When given, the rows gain an n_occurrences column (NA for a key the snapshot does not hold).
verbose	Logical.

Value

A normalized data.frame ready for `precompute_backbone()`.

normalize_wcvp	<i>Normalize one block of WCVp rows to the unified schema</i>
----------------	---

Description

Split out of [read_wcvp\(\)](#) so the streaming build can apply it to a chunk at a time. Acceptance is read from a comparison of two columns of the same row, so nothing here depends on rows outside the block.

Usage

```
normalize_wcvp(df, verbose = TRUE)
```

Arguments

df	A data.frame of raw WCVp rows.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

normalize_wfo	<i>Normalize one block of WFO rows to the unified schema</i>
---------------	--

Description

Split out of [read_wfo\(\)](#) so the streaming build can apply it to a chunk at a time. Nothing here depends on rows outside the block.

Usage

```
normalize_wfo(df, verbose = TRUE)
```

Arguments

df	A data.frame of raw WFO rows, with the file's own column names, decoded from UTF-8.
verbose	Logical.

Value

A normalized data.frame.

normalize_worms	<i>Normalize one block of WoRMS rows to the unified schema</i>
-----------------	--

Description

Split out of `read_worms()` so the streaming build can apply it to a chunk at a time. Nothing here depends on rows outside the block.

Usage

```
normalize_worms(df, verbose = TRUE)
```

Arguments

df	A data.frame of raw WoRMS rows, with the archive's own column names.
verbose	Logical.

Value

A normalized data.frame.

parse_algae_traits	<i>Parse AlgaeTraits macroalgal traits (WoRMS ZIP export)</i>
--------------------	---

Description

Parse AlgaeTraits macroalgal traits (WoRMS ZIP export)

Usage

```
parse_algae_traits(path)
```

Arguments

path	Character. Directory holding CSV/TXT/XLSX files extracted from the Algae-Traits archive.
------	--

Value

data.frame with canonical_name + algae trait columns.

 parse_alien_first_records

Parse the Seebens et al. global first-record database

Description

Reads the public dataset table and reduces it to one row per species x location. The release's locations are non-overlapping: countries, and islands or other parts of a country recorded as regions of their own. A location that is a whole country is keyed on its ISO 3166-1 alpha-2 code and a sub-national one on its own name (`.seebens_location_key()`), in the `location` column; `country_code` carries the ISO code of the country every location lies in, so a roll-up to countries is an explicit choice made on that column. Folding the parts into their country would date the country's first record from an island's: in v4.0 that moved 665 species x country years earlier (Hawaii alone 337, by a median of 25 years) and credited 3,541 pairs to a country whose own location has no record of the species.

Usage

```
parse_alien_first_records(path)
```

Arguments

`path` Character. Path to the FirstRecords dataset CSV.

Details

Every location the source names must map. The map is keyed on wording, so a release that renames a place would otherwise drop its records and return a smaller table rather than an error: v4.0 alone renamed 24 locations, among them "United States" to "United States of America", which is 8,050 records. A name the map does not hold is therefore a hard error, and the entries that resolve to no single country (a record spanning several, "USACanada") are recorded in the map as NA so they read as known rather than missing.

Where a species has several records for one location the earliest year wins, but a record asserting the species is present is preferred to one recording it as absent, uncertain or captive whatever the years are, so the retained row does not date a country's occurrence from a record denying it. That record's own status is published beside its year as `alien_first_record_status`, since the remaining columns are reduced over every record for the pair and a status reduced that way would describe some record other than the one the year came from. The same earliest-wins reduction (`.keep_earliest_first_record()`) runs again after cross-backbone name resolution, at the accepted-species grain, so synonyms collapsing onto one concept take its earliest observed year rather than an arbitrary one. The two convention markers (`.alien_first_record_sentinels`) are dropped to NA here so neither can win that reduction or be served as a year.

Value

data.frame with `canonical_name` + `location` + `country_code` + first-record cols.

parse_alpine_carabids *Parse the Alpine elevational-gradient carabid trait table*

Description

Traits.csv from the Dryad deposit behind Chamberlain et al.'s study of carabid occurrence along Alpine elevation gradients: 185 species recorded at 416 sites between 697 and 2840 m in the Italian Alps, semicolon-delimited.

Usage

```
parse_alpine_carabids(path)
```

Arguments

path Character. Path to the downloaded Traits.csv.

Details

The widest European carabid trait block after the German table, and the one that reaches the Alpine fauna the lowland compilations miss. Its body sizes share the carabids.org ancestry every European carabid source has (23.9% exactly equal to the German table over 92 shared species), so agreement with those is not corroboration; it runs 1.03 against them, close enough to be the same quantity and far enough from 1.000 not to be a verbatim copy in the way the Eberswalde table is.

The wing column ships as bare letters with no legend, so the codes were read off the data rather than assumed: crossed against the German table's words, b is short-winged (16 of 18), m long-winged (50 of 55) and d dimorphic (18 of 19), which is brachypterous, macropterous and dimorphic.

Value

data.frame with canonical_name, body_length_mm, wing_morph.

parse_amniote *Parse the Amniote Life History Database*

Description

One row per species across birds, mammals and reptiles. The dataset codes missing values as -999; these are converted to NA on every numeric trait.

Usage

```
parse_amniote(path)
```

Arguments

path Character. Path to Amniote_Database_Aug_2015.csv.

Value

data.frame with canonical_name + life-history traits.

parse_amphibio	<i>Parse AmphiBIO amphibian traits (CSV from ZIP)</i>
----------------	---

Description

Parse AmphiBIO amphibian traits (CSV from ZIP)

Usage

```
parse_amphibio(path)
```

Arguments

path Character. Path to the AmphiBIO CSV.

Value

data.frame with canonical_name + trait columns.

parse_anage	<i>Parse AnAge longevity and life-history traits (TSV from ZIP)</i>
-------------	---

Description

Parse AnAge longevity and life-history traits (TSV from ZIP)

Usage

```
parse_anage(path)
```

Arguments

path Character. Path to anage_*.txt.

Value

data.frame with canonical_name + AnAge trait columns.

parse_animaltraits	<i>Parse AnimalTraits observations CSV (aggregate to species medians)</i>
--------------------	---

Description

Parse AnimalTraits observations CSV (aggregate to species medians)

Usage

parse_animaltraits(path)

Arguments

path Character. Path to observations.csv.

Value

data.frame with canonical_name + body_mass_kg + metabolic_rate_w.

parse_arctic	<i>Parse Arctic Traits Database (marine benthos)</i>
--------------	--

Description

Long-format fuzzy-coded traits (taxon x trait x category affinity); each trait is reduced to its dominant category per species, using the database’s own category labels.

Usage

parse_arctic(path)

Arguments

path Path to the Arctic Traits CSV.

Value

data.frame with canonical_name + benthic traits.

 parse_arthropod_traits

Parse NW European Arthropod DwC-A (taxon + measurement + description)

Description

Parse NW European Arthropod DwC-A (taxon + measurement + description)

Usage

```
parse_arthropod_traits(dir_path)
```

Arguments

dir_path Character. Directory holding the DwC archive.

Value

data.frame with canonical_name + arthropod trait columns.

 parse_austraits

Parse the AusTraits plant trait database

Description

Reads the long traits.csv from the plain-text release and pivots a set of high-coverage functional traits to one row per taxon, aggregating across all records (numeric by median, categorical by mode). Numeric traits such as leaf mass per area are recorded mostly at population/individual level, so records of every entity type are aggregated up to the taxon.

Usage

```
parse_austraits(path)
```

Arguments

path Character. Path to the extracted release directory or to traits.csv.

Details

Every trait column <col> has a <col>_source column naming the AusTraits datasets (dataset_id) behind the value: for a categorical trait the datasets whose records state the reported value, for a numeric trait every dataset whose records enter the median. The ids resolve against the reference table attached with [attach_references\(\)](#), built from the release's methods.csv (each dataset's primary source citation) and sources.bib (its DOI).

Value

data.frame with canonical_name + plant traits.

parse_avonet	<i>Parse AVONET bird morphology XLSX</i>
--------------	--

Description

Parse AVONET bird morphology XLSX

Usage

parse_avonet(path)

Arguments

path Character. Path to AVONET_BirdLife.xlsx.

Value

data.frame with canonical_name + morphology columns.

parse_bacdive	<i>Parse the BacDive strain snapshot into per-species microbial traits</i>
---------------	--

Description

Reads the gzipped JSONL of full strain records, extracts phenotypic and growth-condition traits per strain, and aggregates to one row per species (categorical by mode, numeric by median). Temperature and pH prefer the optimum measurement, falling back to the growth measurement; ranges are reduced to their midpoint.

Usage

parse_bacdive(path)

Arguments

path Character. Path to bacdive.jsonl.gz.

Value

data.frame with canonical_name + microbial trait columns.

parse_baseflor	<i>Parse Baseflor (Catminat / Julve) French flora trait spreadsheet</i>
----------------	---

Description

Reads the baseflor sheet of Julve's baseflor.xlsx, builds a clean binomial canonical_name from the nomH/nomB/nomA columns, splits the floraison flowering-period field into begin/end months, and recodes the French categorical traits (pollination vector, dispersal mode, breeding system, flower colour, fruit type, woody growth form) to English. Also keeps the two Ellenberg-style axes absent from EIVE: continentality and salinity.

Usage

```
parse_baseflor(path)
```

Arguments

path	Character. Path to the downloaded baseflor.xlsx.
------	--

Details

Indicator values L/T/M/R/N are intentionally not emitted here (the European-calibration EIVE enrichment already covers them); Raunkiaer life form is left to the leda enrichment for the same flora; maximum vegetative height is omitted because the source column is empty.

Value

data.frame with canonical_name + 10 trait columns.

parse_bee_ostwald	<i>Parse Ostwald global bee morphometrics</i>
-------------------	---

Description

Long-format Darwin Core measurement table; reduced to one row per species. verbatimAcceptedNameUsage carries authorship and subgenera, so the join key is reduced to a clean binomial.

Usage

```
parse_bee_ostwald(path)
```

Arguments

path	Path to the morphological dataset CSV.
------	--

Value

data.frame with canonical_name + bee morphometrics.

parse_bet	<i>Parse the Bryophytes of Europe Traits (BET) table</i>
-----------	--

Description

One row per species. The source is R write.table output (space-separated, quoted strings, leading row-id column). Ellenberg-style indicator values use "x" for indifferent, which is mapped to NA. The six one-hot substrate class flags are also collapsed into a pipe-delimited substrate set (e.g. "soil|rock|bark") over soil, rock, bark, wood, living_plants and dung_carcass.

Usage

parse_bet(path)

Arguments

path Character. Path to betdata.txt.

Value

data.frame with canonical_name + bryophyte traits.

parse_betsi_collembola_body_length	<i>Parse the BETSI Collembola body-length export into per-species values</i>
------------------------------------	--

Description

Reads the Zenodo BETSI body-length export and reduces its per-source measurements to one row per Collembola species. bets_i_body_length_mm is the median of the recorded measurements (mm); bets_i_body_length_min_mm/bets_i_body_length_max_mm bound them; bets_i_body_length_n and bets_i_body_length_sources are the number of measurements and of distinct literature sources behind the species' value.

Usage

parse_betsi_collembola_body_length(path)

Arguments

path Character. Path to the BETSI body-length .xlsx (or a directory containing it).

Value

data.frame with canonical_name and the bets_i_body_length_* columns, one row per species.

parse_betsi_recovery	<i>Parse a per-taxon BETSI-recovery matrix into per-species trait columns</i>
----------------------	---

Description

Reads the frozen matrices for the sources feeding enrichment and returns one row per species. Fuzzy sources (species, trait, class, pct) are checked for the fuzzy-coding invariant (each species-by-trait affinity block sums to 100, tolerating +/-1 from integer rounding) and pivoted to one numeric column per <trait>__<modality> bin (0-100). Hard sources (one value per trait) are read into one column per trait. Provenance is not a data column here; it is written to meta.json from the descriptor by the build.

Usage

```
parse_betsi_recovery(enrichment, path)
```

Arguments

enrichment	Character. A per-taxon recovery enrichment name (see list_betsi_recovery()).
path	Character. Directory holding the source .csv files (staged from inst/extdata/betsi/ by the build), or a single .csv when the enrichment has one source.

Value

data.frame with canonical_name and the trait columns, one row per species.

See Also

[build_betsi_recovery\(\)](#)

parse_beukhof	<i>Parse Beukhof marine fish traits</i>
---------------	---

Description

Per-region records (one row per species x marine region) reduced to species-level values (numeric by median, categorical by mode).

Usage

```
parse_beukhof(path)
```

Arguments

path	Path to the Beukhof trait xlsx.
------	---------------------------------

Value

data.frame with canonical_name + marine-fish traits.

parse_bien	<i>Parse BIEN trait data (queried via the BIEN R package)</i>
------------	---

Description

Like fishbase via rfishbase, BIEN is queried directly: a single bulk pull of the selected traits across all species (one query per trait list), filtered to public-access records, then pivoted to one row per species (numeric by median, categorical by mode). The path argument is unused (kept for a uniform parser interface). Curated numeric columns are checked against the unit each BIEN record carries and converted to the unit in the column name (leaf dry mass is recorded in g and written in mg).

Usage

```
parse_bien(path)
```

Arguments

path	Unused.
------	---------

Value

data.frame with canonical_name + plant traits.

parse_birdbase	<i>Parse the BIRDBASE bird trait database</i>
----------------	---

Description

Reads the Data sheet (real header is the second physical row). Presence-coded columns (island, restricted-range) use a blank for absent, mapped to 0. Traits redundant with AVONET morphology are not carried. One row per species.

Usage

```
parse_birdbase(path)
```

Arguments

path	Character. Path to the BIRDBASE xlsx.
------	---------------------------------------

Value

data.frame with canonical_name + bird traits.

parse_blanchard	<i>Parse Blanchard & Moreau ant genus defensive traits</i>
-----------------	--

Description

Genus-level ant traits with codes defined in the source headers; the clearly defined ones are mapped to their labels. Columns are read by position because the source header cells embed long code legends.

Usage

```
parse_blanchard(path)
```

Arguments

path	Path to the ant traits xlsx (header on row 3).
------	--

Value

data.frame with canonical_name (genus) + ant traits.

parse_brot	<i>Parse BROT 2.0 Mediterranean plant traits</i>
------------	--

Description

Long-format trait records reduced to one row per species. Numeric units follow the standardised BROT 2.0 conventions (seed mass mg, SLA mm²/mg, height m, leaf area mm²).

Usage

```
parse_brot(path, sources = file.path(dirname(path), "brot_sources.csv"))
```

Arguments

path	Path to BROT2_dat.csv.
sources	Path to BROT2_sou.csv. Defaults to brot_sources.csv beside path, where the registry download places it.

Details

Every trait column <col> has a <col>_source column naming the BROT sources (SourceID) behind the value: for a categorical trait the sources whose records state the reported value, for a numeric trait every source whose records enter the median. The ids resolve against BROT's own sources file (BROT2_sou.csv: ID, FullSource), attached with [attach_references\(\)](#).

Value

data.frame with canonical_name + plant traits.

parse_bryoatt	<i>Parse BRYOATT (attributes of British and Irish bryophytes)</i>
---------------	---

Description

One row per moss, liverwort or hornwort taxon. Retained traits are the five Ellenberg indicator values (light, moisture, reaction, nitrogen, salt), and the source's life-form, plant-group (moss / liverwort / hornwort) and status codes (kept verbatim).

Usage

```
parse_bryoatt(path)
```

Arguments

path Path to Bryoatt_updated_2017.xls (or its directory).

Value

data.frame with canonical_name + British/Irish bryophyte attributes.

parse_ccdb	<i>Parse the Chromosome Counts Database (CCDB)</i>
------------	--

Description

Somatic chromosome numbers for plants, taken from the CCDB statistics service (one server-side per-taxon reduction per major plant group). Names are collapsed to the binomial (infraspecific and authorship qualifiers are dropped) and reduced to one row per species: the median chromosome number (2n) with its minimum and maximum across the aggregated records. The service reports gametic numbers, which are doubled to the somatic 2n the columns name; the spread between minimum and maximum is real ploidy variation among a species' cytotypes.

Usage

```
parse_ccdb(path)
```

Arguments

path Directory of per-group CCDB statistics CSVs (or a single CSV).

Value

data.frame with canonical_name + chromosome-number traits.

parse_cefas_btrait	<i>Parse the Cefas biological-traits database (North-West European benthos)</i>
--------------------	---

Description

Fuzzy-coded biological traits of North-West European benthic invertebrates, distributed as a flat one-row-per-taxon matrix at genus rank and above. Each of the ten biological traits is split across several modality columns holding a 0-3 fuzzy score (0 = no evidence ... 3 = strong evidence); each trait is collapsed to the single modality with the highest score. The traits are body size (mm class), morphology, lifespan (year class), egg development, larval development, living habit, sediment position, feeding mode, mobility and bioturbation mode. The WoRMS AphiaID is present in the source but the Genus-rank name is used as the resolution key.

Usage

```
parse_cefas_btrait(path)
```

Arguments

path	Path to the Trait Matrix CSV (or a directory containing it).
------	--

Value

data.frame with canonical_name + benthic biological traits.

parse_chelonians	<i>Parse the CheloniansTraits turtle database</i>
------------------	---

Description

Reads the data sheet (real header is the second physical row), treats the literal "NA" as missing, and parses "min-max" text ranges to their midpoint. One row per species.

Usage

```
parse_chelonians(path)
```

Arguments

path	Character. Path to CheloniansTraits dataset.xlsx.
------	---

Value

data.frame with canonical_name + turtle traits.

parse_chowdhury	<i>Parse the Chowdhury et al. 2025 German carabid trend + trait table</i>
-----------------	---

Description

Supporting Information Data S1 of "Widespread Decline of Ground Beetles in Germany" (Diversity and Distributions, doi:10.1111/ddi.70112). One row per species: 383 rows, of which 382 carry trait values (*Ocys tachysoides* has none), which makes this by a wide margin the largest lawfully reachable carabid trait block.

Usage

parse_chowdhury(path)

Arguments

path Character. Path to the downloaded .xlsx.

Details

Trait values are kept verbatim rather than normalized here. The source vocabularies are small and self-describing, and taxify’s cross-source trait registry is where a vocabulary is mapped, so the .vtr keeps what the authors wrote and the registry decides what it means.

Read off the file rather than the paper: trophicLevel has three values (Predator 274, Herbivore 80, Omnivore 28) and no mycetophagae category, and habitatPref has seven classes, not eight.

Alongside the traits the table carries the paper’s own result, a modelled two-year occupancy trend per species. Nothing else in taxify holds a national population trend, so those columns are kept.

Value

data.frame with canonical_name plus trait and trend columns.

parse_clopla	<i>Parse CLO-PLA (clonal and bud-bank traits of the Central European flora)</i>
--------------	---

Description

One row per species for 29 traits of clonal growth, bud banks and lifespan. Names are collapsed to the binomial (authorship and infraspecific rank are dropped, hybrid markers kept), so infraspecific records are aggregated to the species. Numeric traits (growth-form flags, bud-bank sizes/depths, persistence, offspring, spread, clonal index) are reduced by median; the nominal coded traits (branching, cyclicity, clonal-growth-organ type, regenerative-bud position and role) by mode. Coded fields are kept verbatim as in the source; consult the CLO-PLA metadata to decode them.

Usage

```
parse_clopla(path)
```

Arguments

path Path to CLO-PLA-traits.txt (or a directory containing it).

Value

data.frame with canonical_name + CLO-PLA clonal / bud-bank traits.

parse_combine	<i>Parse the COMBINE mammal trait database (reported values)</i>
---------------	--

Description

Uses the reported (observed/compiled) trait table, not the phylogenetically imputed one, so gap-filled numbers are never shipped as observed. Keyed on the IUCN 2020 binomial.

Usage

```
parse_combine(path)
```

Arguments

path Character. Path to trait_data_reported.csv.

Value

data.frame with canonical_name + mammal traits.

parse_common_names	<i>Parse common names from GBIF, NCBI, and OTT</i>
--------------------	--

Description

Merges vernacular names from three sources:

- GBIF: VernacularName.tsv (has ISO 639-1 language codes)
- NCBI: names.dmp where name_class == "common name" (no language)
- OTT: synonyms.tsv where type == "common name" (no language)

Usage

```
parse_common_names(dir_path)
```

Arguments

dir_path Character. Directory containing gbif/, ncbi/, ott/ subdirs.

Details

NCBI and OTT common names have no language tag, so lang is set to NA.

Value

data.frame with canonical_name + lang + common_name.

parse_copepod_traits *Parse the Brun et al. marine copepod trait database*

Description

Species-level functional traits of marine copepods, distributed as a multi-sheet PANGAEA workbook (one sheet per trait family). Each trait sheet is melted to (taxon, trait, value); the binary feeding-mode and spawning-strategy flags are first collapsed to a single categorical label per record. Records are then reduced to one row per taxon (numeric traits by median, categorical by mode). Subgenus qualifiers in parentheses (*Acartia* (*Acartiura*) *clausi*) are stripped so the name matches a backbone binomial or genus. Retained traits: body length (mm), egg outer diameter (micrometres), clutch size (egg count), feeding mode, feeder type, spawning strategy, presence of myelination and of resting eggs.

Usage

```
parse_copepod_traits(path)
```

Arguments

path Path to the Brun-et-al_2016-Copepode_trait.xlsx workbook.

Value

data.frame with canonical_name + copepod traits.

parse_coral_traits	<i>Parse the Coral Trait Database release</i>
--------------------	---

Description

The release is long format (one row per observation). This pivots a set of high-coverage functional traits to one row per species, aggregating numeric traits by median and categorical traits by mode.

Usage

```
parse_coral_traits(path)
```

Arguments

path	Character. Path to the extracted release directory or to ctdb_1.1.1_data.csv.
------	---

Value

data.frame with canonical_name + coral trait columns.

parse_diaz_traits	<i>Parse Diaz et al. 2022 supplementary traits (XLSX)</i>
-------------------	---

Description

Parse Diaz et al. 2022 supplementary traits (XLSX)

Usage

```
parse_diaz_traits(path)
```

Arguments

path	Character. Path to the Diaz 2022 XLSX.
------	--

Value

data.frame with canonical_name + seed_mass_mg + plant_height_m.

parse_disperse	<i>Parse DISPERSE European aquatic-invertebrate dispersal traits</i>
----------------	--

Description

Genus-level fuzzy-coded traits. Each fuzzy trait group (body size, life cycle, reproductive cycles, dispersal strategy, adult life span, female wing length, wing-pair type, fecundity) is reduced to its dominant modality, labelled with the database's own modality descriptions. Modality labels are ASCII-normalized (<=, >=, -), and the three binned physical-magnitude ordinals (disperse_body_size_cm, disperse_female_wing_mm, disperse_fecundity) also get a coarse _mid numeric-midpoint column for numeric joins; the count/time ordinals stay categorical.

Usage

```
parse_disperse(path)
```

Arguments

path	Path to the DISPERSE.xlsx.
------	----------------------------

Value

data.frame with canonical_name (genus) + dominant-modality traits + the three _mid midpoint columns.

parse_eberswalde	<i>Parse the Eberswalde long-term carabid monitoring trait table</i>
------------------	--

Description

EWcarabids1999-2022_species_trends_traits.csv from the Leuphana deposit behind the 24-year drought study of Weiss, von Wehrden and Linde: 27 species from 13 forest plots near Eberswalde, semicolon-delimited with German decimal commas.

Usage

```
parse_eberswalde(path)
```

Arguments

path	Character. Path to the downloaded .csv.
------	---

Details

Two of its columns are carabids.org verbatim, which the deposit's own README states outright ("size = mean size (body length, mm) ... Source: Homburg et al. (2014) / carabids.org", likewise wings and latitude). The data confirm it: over the 19 species shared with the Chowdhury table every single size is exactly equal and every wing class agrees. They are kept because a door should surface what its source carries, and they are excluded from the cross-source trait registry, where they would double-count one lineage.

What belongs to this deposit alone: a 24-year local abundance trend, a sensitivity to the 72-month SPEI drought index, and a feeding guild refined by the authors' field observations to name the prey (Gastropoda, Collembola). The humidity preference is Sustek's (2004) 1-8 scale, an independent source.

Value

data.frame with canonical_name plus trait columns.

parse_ecoflora	<i>Parse the Ecoflora scrape snapshot (British Isles plant traits)</i>
----------------	--

Description

Reads the per-species Ecoflora scrape (results.csv, one row per species, trait columns named by their Ecoflora short codes) and returns a clean wide data.frame keyed on canonical_name. Numeric fields (heights, seed weight, flowering months) take the median of any multiple scraped values; categorical fields keep the unique values joined with "; ". Ecoflora records height in cm ("Height: 3. typical maximum (cm)", "Height: 4. typical minimum (cm)") and seed weight in mg ("Seed weight: 3. mean (mg)"); heights are written in mm, seed weight as recorded. Every trait column carries a _uk suffix to mark the British-flora calibration and to avoid collisions when chained with other plant-trait enrichments.

Usage

```
parse_ecoflora(path)
```

Arguments

path	Character. Path to the Ecoflora results.csv snapshot.
------	---

Details

Ecoflora (Fitter & Peat 1994) has no bulk download or API; the snapshot was collected one species at a time and is redistributed under the source licence (CC BY-NC-SA 4.0).

Value

data.frame with canonical_name + 18 _uk trait columns.

parse_ecomorphosis	<i>Parse the Bonfanti et al. Collembola ecomorphosis list into per-species rows</i>
--------------------	---

Description

Reads the Zenodo ecomorphosis species list and returns one row per species known to display ecomorphosis.

Usage

```
parse_ecomorphosis(path)
```

Arguments

path Character. Path to Annex_ASE_Online_db.xlsx (or a directory containing it).

Details

Columns:

- ecomorphosis – TRUE for every listed species (a presence list).
- ecomorphosis_area – geographical area the record applies to.
- ecomorphosis_reference – the literature record establishing it.

Value

data.frame with canonical_name and the ecomorphosis* columns, one row per species.

parse_edwards_phyto	<i>Parse Edwards phytoplankton nutrient-utilization traits</i>
---------------------	--

Description

Per-culture nitrogen (nitrate, ammonium) and phosphorus utilization traits (maximum growth rate, half-saturation constants for growth and uptake, minimum subsistence quota, maximum quota, maximum uptake rate) from Edwards et al. (2015, Ecological Archives E096-202). Each source row is one culture measurement; rows are reduced to one per species (numeric traits by median, taxon group and freshwater/marine system by mode). Experimental conditions (temperature, irradiance, daylength) and per-measurement citations are not carried. Nutrient-trait column names are kept verbatim from the source: units are documented in the source metadata, and the source is internally inconsistent on the V_max time unit, so no unit is asserted in the name.

Usage

```
parse_edwards_phyto(path)
```

Arguments

path Directory holding Table1.csv (or a path to Table1.csv itself).

Value

data.frame with canonical_name + phytoplankton nutrient traits.

parse_eive	<i>Parse EIVE 1.0 ecological indicator values (XLSX)</i>
------------	--

Description

Parse EIVE 1.0 ecological indicator values (XLSX)

Usage

```
parse_eive(path)
```

Arguments

path Character. Path to EIVE_1.0.xlsx.

Value

data.frame with canonical_name + indicator columns.

parse_ellers_collembola	<i>Parse the Ellers et al. Collembola trait table into per-species values</i>
-------------------------	---

Description

Reads the CC0 Collembola_trait_data.csv (Ellers et al. 2018) and returns one row per species. Ordinal codes whose levels are named in the source are decoded (reproduction, moisture preference); the biogeographic temperature-zone class and the thermal-niche-breadth count are kept as their source ordinal codes.

Usage

```
parse_ellers_collembola(path)
```

Arguments

path Character. Path to Collembola_trait_data.csv (or a directory containing it).

Details

Columns:

- `ellers_vertical_distribution` – soil, subsurface or surface.
- `ellers_body_size_mm` – body size (mm).
- `ellers_reproduction` – asexual or sexual.
- `ellers_moisture_pref` – xerophilous, xero-mesophilous, mesophilous, hygro-mesophilous or hygrophilous.
- `ellers_temperature_pref` – biogeographic temperature-zone class, ordinal 1-5 (1 = boreal zone only ... 5 = Mediterranean zone only) per the source codebook.
- `ellers_thermal_niche_breadth` – number of biogeographic zones occupied, ordinal 1-3.

Value

data.frame with `canonical_name` and the `ellers_*` trait columns, one row per species.

<code>parse_elton_traits</code>	<i>Parse EltonTraits 1.0 birds + mammals TSVs</i>
---------------------------------	---

Description

Parse EltonTraits 1.0 birds + mammals TSVs

Usage

```
parse_elton_traits(birds_path, mammals_path)
```

Arguments

<code>birds_path</code>	Character. Path to BirdFuncDat.txt.
<code>mammals_path</code>	Character. Path to MamFuncDat.txt.

Value

data.frame with `canonical_name` + diet/foraging/body mass columns.

parse_epa_freshwater	<i>Parse the EPA Freshwater Biological Traits database (macroinvertebrates)</i>
----------------------	---

Description

Ecological and life-history traits of North-American freshwater macroinvertebrates, read from the transposed distribution (one row per taxon and source citation). A curated set of interpretable primary trait columns is melted to (taxon, trait, value) and reduced to one row per taxon by mode, so the multiple per-taxon citation records collapse to the dominant state. The retained traits are primary feeding mode, habit, voltinism, thermal preference, maximum body-size class, body shape, rheophily, primary oviposition behaviour and diapause. Other (specify in comments) placeholder entries are dropped. The taxon grain is mixed (species and genus); the ITIS TSN is present in the source. This dataset shares its lineage (USGS Data Series 187, Vieira et al. 2006) with freshwater_insects_conus, which is at genus grain; the species-grain records here are the net-new contribution.

Usage

```
parse_epa_freshwater(path)
```

Arguments

path	Path to the transposed traits .xls (or its directory).
------	--

Value

data.frame with canonical_name + freshwater macroinvertebrate traits.

parse_eupolltrait	<i>Parse EuPollTrait (European bees + hoverflies)</i>
-------------------	---

Description

Long-format Darwin Core measurement-or-fact table joined to the taxon core (genus + specific epithet) for clean binomials; reduced to one row per species for a curated set of morphological, biogeographic and ecological traits.

Usage

```
parse_eupolltrait(path)
```

Arguments

path	Directory holding mof.csv and taxon.csv.
------	--

Value

data.frame with canonical_name + pollinator traits.

parse_eurobat	<i>Parse the EuroBaTrait European bat trait dataset</i>
---------------	---

Description

Species-level traits of European bats spread across thematic Darwin Core measurement-or-fact tables (morphology, life history, diet, foraging habitat, roost type). Each table is melted to (species, trait, value), stacked, and reduced to one row per species (numeric traits by median, categorical by mode). Underscored source names (Genus_species) are normalised to a binomial.

Usage

parse_eurobat(path)

Arguments

path Directory holding the thematic trait CSVs.

Value

data.frame with canonical_name + European bat traits.

parse_euromed_distribution	<i>Parse the Euro+Med distribution snapshot</i>
----------------------------	---

Description

Reads euromed_distribution.jsonl and emits one row per (accepted taxon, Euro+Med area): the status class, the term as the portal states it, the area with its ISO 3166-1 alpha-2 country code where the area is one country, the taxon's CDM UUID and the ids of the references behind the reported status. A retracted report (reported in error) is dropped; a former presence is extinct. Where records of one taxon-area disagree the status follows .euromed_status_order and the references are those stating it.

Usage

parse_euromed_distribution(path)

Arguments

path Character. Path to euromed_distribution.jsonl.

Details

Areas whose name is not in .euromed_area_iso and is not a combined area or part of a country are reported so the table can be extended.

Value

data.frame with canonical_name, area_code, area_name, area_level, iso2, euromed_status, euromed_status_detail, euromed_status_source and taxon_id, with a reference table (ref_id, citation, doi) attached by [attach_references\(\)](#).

 parse_faprotax

Parse FAPROTAX, the functional annotation of prokaryotic taxa

Description

FAPROTAX.txt from the release zip (Louca et al. 2016, Science 353:1272-1277): 92 metabolic and ecological function groups, each listing the prokaryotic taxa known to perform it, compiled from IJSEM and Bergey's Manual.

Usage

```
parse_faprotax(path)
```

Arguments

path Character. Path to the unpacked release directory or to FAPROTAX.txt itself.

Details

The file is a grouped list rather than a table. A group begins with its name plus a tab-separated attribute string; the lines under it are member taxa written as taxonomic path suffixes (*Proteobacteria*Nitrosomonas*, *Escherichia*coli*). A group may also inherit or exclude another group's members through add_group: / subtract_group:, which are resolved here in file order – a referenced group is always defined earlier.

Members are annotated at whatever rank the evidence supports, so the file mixes species, genera and higher taxa in one list. taxify joins an enrichment on a species name or a genus, so each member is reduced to whichever of those it carries: the last genus-plus-epithet pair in the path if there is one, else the last plain capitalised name. Strain designations (107, 4_3_47FAA, 2.4.3) carry neither and are dropped. Rows keyed on a rank above genus are kept as written – they simply never match a genus, at the cost of a few hundred rows.

The result is one row per taxon with the function groups it belongs to as a pipe-delimited set, the same shape the NestTrait modality columns use: a prokaryote genuinely performs several of these, and collapsing to one would be a choice the source does not make.

Value

data.frame with canonical_name, faprotax_functions, faprotax_n_functions.

parse_finand	<i>Parse the Finand & Kotze Helsinki urban-forest carabid trait table</i>
--------------	---

Description

Species_traits_zenodo.xlsx from the Zenodo deposit behind Finand & Kotze's urban forest fragmentation study: 34 species from 25 remnant forests in Helsinki, with body length measured from the specimens the authors caught rather than compiled from a database.

Usage

```
parse_finand(path)
```

Arguments

path	Character. Path to Species_traits_zenodo.xlsx.
------	--

Details

That independence is the reason to carry a table this small. Every other reachable carabid trait source traces back to carabids.org (Homburg et al. 2014), so none of them can check another. This one can: its body lengths run at a median ratio of 0.9946 against arthropod_traits over 28 shared species (IQR 0.985-1.008) with only 7.1 percent exactly equal, and at ~1.00 against the Chowdhury table over 14. Values such as 6.066667 mm are means over individuals, not a rounded lookup.

Body_length is a whole-body length, not the elytra length the name might suggest: elytra would sit near 0.6 of these values, and the published body lengths agree directly (Cychrus caraboides 16.9 mm against 14-19 in the literature, Carabus glabratus 28.6 against 22-34).

The wing column is the same three-state trait the German table carries. On the 25 species they share there is no long-versus-short disagreement at all, but seven species the German table calls dimorphic get a definite morph here, because this records the morph of the beetles actually caught in Helsinki while the German table states the species' capacity.

Value

data.frame with canonical_name plus trait columns.

parse_fishbase	<i>Parse FishBase species traits (via rfishbase)</i>
----------------	--

Description

Parse FishBase species traits (via rfishbase)

Usage

```
parse_fishbase(path)
```

Arguments

path Character. Not used (rfishbase fetches data directly), kept for interface consistency.

Value

data.frame with canonical_name + body/depth/diet columns.

parse_fishmorph	<i>Parse FISHMORPH freshwater fish morphological traits (CSV)</i>
-----------------	---

Description

Parse FISHMORPH freshwater fish morphological traits (CSV)

Usage

```
parse_fishmorph(path)
```

Arguments

path Character. Path to FISHMORPH_Database.csv.

Value

data.frame with canonical_name + morphology columns.

parse_fishtraits	<i>Parse the FishTraits v14 database (Frimpong & Angermeier)</i>
------------------	--

Description

Species-level traits of United States freshwater fishes, one row per species in a legacy Excel workbook. The table is already at the species grain, so it is read wide and a curated, documented subset of columns is retained: the binary trophic-guild indicators, life-history quantities (maximum total length in cm, age at maturity and longevity in years, maximum fecundity, spawning-season length), salinity and temperature tolerances (30-year mean January minimum and July maximum at the range centroid, in degrees C), and conservation flags. The 25 Balon reproductive-guild indicator columns are collapsed to a single repro_guild code (the flagged guild). The ITIS TSN is carried as a cross-reference. Field meanings follow the dataset's AttributeDefinitions.

Usage

```
parse_fishtraits(path)
```

Arguments

path Path to FishTraits_14.3.xls (or a directory containing it).

Value

data.frame with canonical_name + US freshwater-fish traits.

parse_floraweb	<i>Parse the FloraWeb scrape snapshot (German-flora plant traits)</i>
----------------	---

Description

Reads the long-format FloraWeb scrape (results_long.csv, one row per species/page/label/value) and pivots it to a clean wide data.frame keyed on canonical_name. German "no data" placeholders are dropped, multiple values for one trait are collapsed to unique values joined with "; ", and the German trait labels are mapped to English column names (each with a _de suffix). Covers morphology, reproductive biology, the nine Ellenberg indicator values, ploidy and chromosome number, and chorological distribution.

Usage

parse_floraweb(path)

Arguments

path Character. Path to the FloraWeb results_long.csv snapshot.

Details

FloraWeb (Bundesamt fuer Naturschutz) is the live portal carrying the BiolFlor trait data (Klotz, Kuehn & Durka 2002) plus Rothmaler morphology and Ellenberg indicator values. It has no bulk export or API, so the snapshot was scraped per species; the access date is its version.

Value

data.frame with canonical_name + the mapped _de trait columns.

parse_frugivoria	<i>Parse Frugivoria (Neotropical frugivore traits)</i>
------------------	--

Description

Combines the "simple" mammal and bird tables on a shared core (diet, body size/mass, longevity, generation time) with a taxon_group discriminator.

Usage

```
parse_frugivoria(path)
```

Arguments

path	Directory holding mammal.csv and bird.csv.
------	--

Value

data.frame with canonical_name + traits.

parse_fungal_traits	<i>Parse FungalTraits XLSX (Table S1, genus-level traits)</i>
---------------------	---

Description

Reads the Polme et al. (2020) supplementary XLSX, selects the most informative trait columns, and returns a data.frame keyed on genus.

Usage

```
parse_fungal_traits(path)
```

Arguments

path	Character. Path to the downloaded XLSX file.
------	--

Value

data.frame with genus + 9 trait columns.

parse_fungalroot	<i>Parse the FungalRoot database (GBIF Darwin Core Archive) to genus-level types</i>
------------------	--

Description

Reads the FungalRoot occurrence core (occurrences.csv) and its MeasurementOrFact extension (measurements.csv), keeps the per-observation Mycorrhiza type measurements, standardizes them to a small set of type tokens, and reduces them to one row per plant genus by majority consensus. Mycorrhizal type is phylogenetically conserved at the genus level, which is the resolution FungalRoot itself recommends for inference; the per-genus value here is the most frequent standardized type across that genus's observations (taxifydb's own aggregation, not FungalRoot's published per-genus assignment).

Usage

```
parse_fungalroot(path)
```

Arguments

path	Character. Directory holding the extracted FungalRoot DwC-A.
------	--

Value

data.frame keyed on genus (also copied to canonical_name) with mycorrhizal_type, mycorrhizal_status, and mycorrhizal_records.

parse_funguild	<i>Parse FUNGuild JSON database dump</i>
----------------	--

Description

Reads the JSON array returned by the FUNGuild API, filters to genus and species-level entries, and returns a clean data.frame with canonical_name and trait columns.

Usage

```
parse_funguild(path)
```

Arguments

path	Character. Path to the downloaded JSON/HTML file.
------	---

Value

data.frame with canonical_name + trophic_mode + guild + growth_morphology + confidence_ranking.

`parse_fw_insects_conus`*Parse the Freshwater Insects CONUS genus trait table (EDI edi.481)*

Description

Genus-level ecological and life-history traits of North American freshwater insects, distributed in long form (one row per genus and trait group). Pivoted to one row per genus; every trait group is categorical (the source ships the abbreviated modality codes, kept verbatim).

Usage

```
parse_fw_insects_conus(path)
```

Arguments

`path` Path to the "Insect Traits by Genus" CSV.

Value

data.frame with `canonical_name` (genus) + freshwater-insect traits.

`parse_gbif_common_names`*Parse GBIF vernacular names (VernacularName.tsv + Taxon.tsv)*

Description

Parse GBIF vernacular names (VernacularName.tsv + Taxon.tsv)

Usage

```
parse_gbif_common_names(dir_path)
```

Arguments

`dir_path` Character. Directory holding VernacularName.tsv and Taxon.tsv.

Value

data.frame with `canonical_name` + `lang` + `common_name`.

parse_gidias	<i>Parse the GIDIAS impact database into per-species EICAT/SEICAT aggregates</i>
--------------	--

Description

Reads the GIDIAS machine-readable impact records and reduces them to per-species environmental- and socio-economic-impact indicators. The `.vtr` carries these derived aggregates; for the individual impact records, use the GIDIAS figshare download.

Usage

```
parse_gidias(path)
```

Arguments

path	Character. Path to the GIDIAS machine-readable <code>.csv</code> (or a directory containing it).
------	--

Details

`gidias_eicat_category` is the species' EICAT category: the most severe magnitude among its negative (harmful) environmental-impact records, mapped from the GIDIAS 0-3 magnitude scale to the IUCN levels MC/MN/MO/MR/MV (the per-record global-extinction flag splits magnitude 3 into local MR and global MV); a species with negative records but no scored magnitude is "DD". `gidias_seicat_category` applies the same rule to the constituents-of- well-being block (SEICAT, MC/MN/MO/MR). The remaining columns summarise the evidence: the driving mechanism(s), affected well-being constituents, the species' functional group and realms, and record/source counts.

Nature's Contributions to People (NCP) is reported as a direction rather than a category: `gidias_ncp_direction` gives the direction(s) recorded ("Negative", "Positive", "Neutral", or a combination for the 16% of species with a mixed record) and `gidias_ncp_affected` the contributions affected. GIDIAS scores no magnitude scale for this block, so no IUCN-style category is derived; the two columns are not paired, so use the source records where the direction of a specific contribution matters.

Each species is aggregated twice. `affected_taxon = "Any"` summarises every record, and is the grain to use unless you want one affected group. The other rows summarise the species' impacts on one affected native taxon ("Plant", "Invertebrate", "Vertebrate", "Microbe", "Fungi"), which 28% of species split across two or more. Only the "Any" row carries SEICAT and the negative records with no affected taxon recorded: the affected-taxon axis slices the environmental-impact block, so the SEICAT columns are NA on a per-taxon row.

Value

data.frame with `canonical_name`, `affected_taxon`, and the `gidias_*` aggregate columns.

 parse_gift

Parse GIFT species-level plant traits (via the GIFT package)

Description

Downloads GIFT's full catalogue of species-level traits from the live API (batched over all trait IDs) and pivots them to one row per species. GIFT aggregates records to a single value per species (mean for numeric traits, most frequent entry for categorical ones). Only the redistributable subset the API returns is kept; restricted references are excluded by the default call. The species name is GIFT's `work_species`.

Usage

```
parse_gift(path = NULL, agreement = 0.66, batch_size = 12L, verbose = TRUE)
```

Arguments

<code>path</code>	Ignored; the GIFT package fetches data directly. Present so the interface matches the file-based parsers.
<code>agreement</code>	Numeric in $[0, 1]$. Minimum agreement among source records for a categorical value to be reported. Default 0.66.
<code>batch_size</code>	Integer. Number of trait IDs per API request. Default 12.
<code>verbose</code>	Logical. Default TRUE.

Details

Every trait column `<col>` has a `<col>_source` column holding the GIFT reference ids (`ref_ID`) GIFT lists for the aggregated value (the references field `GIFT::GIFT_traits()` returns beside it). For a categorical trait these are the references that state the reported value; for a numeric trait the references entering the mean. GIFT prefixes some ids with `-` or `+`, a marking its documentation does not define. The sign is dropped from the cell and kept per reference in the table as `listed_negative` / `listed_positive`. A negatively listed reference never appears among the records `GIFT_traits_raw(bias_ref = FALSE)` returns (the tree-only `GlobalTreeSearch` is one), which fits GIFT's `bias_ref` flag. The reference table comes from `GIFT::GIFT_references()`; an id GIFT lists but does not serve there has NA citation and a note saying so. It is attached with `attach_references()`.

Value

data.frame keyed on `canonical_name` with one `gift_<trait>` column per GIFT trait. Numeric traits are doubles; the rest are character.

parse_globi	<i>Parse the GloBI distinct-edge snapshot into per-species interaction degree</i>
-------------	---

Description

Reads the crawled distinct (source, type, target) edges, resolves both endpoints to their cross-backbone accepted names, and counts, per accepted species, the number of distinct interaction partners (undirected), distinct interaction types, and total interaction records. Degree is a distinct-count, so it is aggregated at the accepted grain (both endpoints resolved) rather than per raw name; the registry entry therefore sets `resolve_names = FALSE`.

Usage

```
parse_globi(path)
```

Arguments

path	Character. Path to <code>globi_edges.tsv.gz</code> .
------	--

Value

data.frame with `canonical_name`, `interaction_degree`, `n_interaction_types`, `n_interaction_records`.

parse_globtherm	<i>Parse the GlobTherm thermal-tolerance database</i>
-----------------	---

Description

One row per species, wide format with two measurement blocks (upper/heat and lower/cold limit), each carrying its own metric definition. The metric columns are kept because a temperature value is ambiguous without them (CTmax, LT50 and upper thermoneutral zone are different quantities).

Usage

```
parse_globtherm(path)
```

Arguments

path	Character. Path to <code>GlobalTherm_upload_*.csv</code> (latin-1 encoded, with duplicate header names handled by column position).
------	---

Value

data.frame with `canonical_name` + thermal-limit traits.

`parse_glonaf`*Parse GloNAF taxon-region data (multiple CSVs/XLSXs from ZIP)*

Description

Parse GloNAF taxon-region data (multiple CSVs/XLSXs from ZIP)

Usage

```
parse_glonaf(dir_path)
```

Arguments

`dir_path` Character. Directory containing the GloNAF files.

Value

data.frame with canonical_name + region_id + naturalized.

`parse_gmpd`*Parse the Global Mammal Parasite Database (GMPD 2.0)*

Description

Host-parasite association records for wild ungulates, carnivores and primates, aggregated to one row per host species. Reported traits are the parasite richness (distinct parasite species), the distinct-parasite count by parasite type (helminth, virus, bacteria, protozoan, arthropod, fungus, prion), the mean reported prevalence and the host group.

Usage

```
parse_gmpd(path)
```

Arguments

`path` Path to GMPD_main.csv (or a directory containing it).

Value

data.frame with canonical_name (host) + parasite-summary traits.

parse_griis	<i>Parse GRIIS Country Compendium CSV</i>
-------------	---

Description

Parse GRIIS Country Compendium CSV

Usage

```
parse_griis(path)
```

Arguments

path Character. Path to GRIIS_Country_Compendium_V1_0.csv.

Value

data.frame with canonical_name + country_code + invasive_status.

parse_groot	<i>Parse the GRooT species-aggregate root-trait table</i>
-------------	---

Description

GRooT ships a long-format CSV (GRooTAggregateSpeciesVersion.csv): one row per species x trait, with per-species mean, median and quartiles. This pivots the nine best-populated key traits to wide format, one row per species, using the per-species mean (meanSpecies). The species name is the TNRS-resolved genusTNRS + speciesTNRS.

Usage

```
parse_groot(path)
```

Arguments

path Character. Path to GRooTAggregateSpeciesVersion.csv.

Value

data.frame keyed on canonical_name with nine root-trait columns.

parse_gwdd*Parse the Global Wood Density Database v2 (species aggregate)*

Description

Reads the species-aggregated file (one row per species). Wood density is reported as wood specific gravity (oven-dry mass / green volume), dimensionless and numerically equal to g/cm³. Bark density is not present in the aggregated files and is not included.

Usage

```
parse_gwdd(path)
```

Arguments

path Character. Path to gwddagg_v2.2_species.csv.

Value

data.frame with canonical_name + wood density traits.

parse_homerange*Parse HomeRange mammal home-range database*

Description

Per-individual home-range records reduced to species medians (home range in km², body mass in kg).

Usage

```
parse_homerange(path)
```

Arguments

path Path to HomeRangeData CSV.

Value

data.frame with canonical_name + home_range_km2 + body_mass_kg.

parse_hosts	<i>Parse NHM HOSTS Lepidoptera hostplant records into per-insect host breadth</i>
-------------	---

Description

Reads the harvested datastore JSONL (one record per line) and rolls up per accepted insect name to the number of distinct hostplant binomials and distinct hostplant families.

Usage

parse_hosts(path)

Arguments

path Character. Path to the hosts.jsonl file.

Value

data.frame with canonical_name, host_plant_count, host_family_count.

parse_huang_amph	<i>Parse Huang amphibian morphology</i>
------------------	---

Description

Three per-order files (Anura, Caudata, Gymnophiona) of per-specimen morphometrics; reduced to species-level medians for the measurements common and comparable across orders (snout-vent length, head length/width, eye diameter, fore/hind-limb length). Order-specific measurements are not carried because column meanings differ between clades.

Usage

parse_huang_amph(path)

Arguments

path Directory holding Anura.csv / Caudata.csv / Gymnophiona.csv.

Value

data.frame with canonical_name + morphometrics + taxon_order.

 parse_hydraulics

Parse the Sanchez-Martinez et al. 2020 plant hydraulics compilation

Description

HydraEvo12020.csv from the figshare deposit behind "Adaptation and coordinated evolution of plant hydraulic traits" (Ecology Letters 23:1599-1610): 2027 seed-plant species with the four traits the paper analyses – xylem specific conductivity (Ks), xylem resistance to embolism (P50), sapwood allocation relative to leaf area (the Huber value Hv) and drought exposure (the minimum midday water potential) – alongside a WorldClim/SoilGrids block describing where each species grows.

Usage

```
parse_hydraulics(path)
```

Arguments

path Character. Path to the downloaded HydraEvo12020.csv.

Details

A genealogy warning that matters for the trait registry: this is a literature compilation, and so are AusTraits and BROt. Where they overlap they often carry the same primary record – 63% of the P50 values shared with AusTraits and 65% of the Ks values are equal to the last digit, and 42% of the P50 values shared with BROt. Agreement between them is therefore not independent corroboration. The overlap is small against what this source adds (74 of its 894 P50 species are in either incumbent), which is why it is registered rather than rejected the way FISHMORPH's max body length was.

The Huber value is the one column needing a conversion, and the overlap is what pins it: 79 of the 208 species shared with AusTraits are equal to the last digit after multiplying by 1e-4, so the source is cm² sapwood per m² leaf where AusTraits is m²/m². The .vtr keeps the source unit and the registry applies the factor.

Only the hydraulic block and the mean annual temperature are kept. The rest of the climate and soil columns describe a site, not the plant.

Value

data.frame with canonical_name plus hydraulic trait columns.

parse_imageomics_neon	<i>Parse the Imageomics NEON ground-beetle image measurements</i>
-----------------------	---

Description

BeetleMeasurements.csv from the 2018-NEON-beetles deposit: 39,064 measurements taken from images of pinned individuals across 86 species at 30 NEON sites, reduced here to a per-species median.

Usage

```
parse_imageomics_neon(path)
```

Arguments

path Character. Path to BeetleMeasurements.csv.

Details

These are North American species measured from specimens, so they are independent of the carabids.org lineage every European carabid source shares, and they reach a fauna the European sources do not cover at all.

The source measures the elytron rather than the whole animal and stores centimetres in dist_cm, so values are converted to the millimetres the elytra_length trait uses. Sanity holds within the source and against it: the length-to-width ratio has a median of 1.79 (range 1.34-2.25), the shape of a carabid elytron, and Carabus nemoralis reads 14.87 mm against the 23.4 mm body length the Finnish specimens give, a ratio of 0.64.

lying_flat = "No" marks a specimen photographed at an angle, where a projected length is fore-shortened; those measurement pairs are dropped.

Value

data.frame with canonical_name plus elytra measurements.

parse_invacost	<i>Parse the InvaCost economic-cost database into per-species aggregates</i>
----------------	--

Description

Reads the InvaCost v4.1 workbook and reduces the individual cost estimates to three per-species economic-impact indicators. The .vtr carries these derived aggregates; for the individual cost estimates, use the invacost package.

Usage

```
parse_invacost(path)
```

Arguments

path Character. Path to InvaCost_database_v4.1.xlsx.

Details

cost_total_usd is the InvaCost-convention cumulative cost: each estimate's standardised annual cost (Cost_estimate_per_year_2017_USD_exchange_rate, in 2017 USD at the exchange rate) is expanded across the number of years its documented impact period spans (Probable_ending_year_adjusted - Probable_starting_year_adjusted + 1, floored at one year where the period is a single year or unrecorded) and summed across every estimate for the species. Temporally or spatially overlapping estimates are not de-duplicated (the invacost package's summarizeCosts() is the tool for a rigorous cost analysis); cost_total_usd is a coarse economic-impact indicator, not an audited figure. cost_type is the merged cost type (damage / management / mixed / unspecified) that accounts for the largest share of the species' cumulative cost.

Value

data.frame with canonical_name, cost_total_usd (2017 USD), cost_n (number of cost estimates), cost_type (dominant cost type).

parse_italic	<i>Parse the ITALIC lichen taxon-page snapshot</i>
--------------	--

Description

Reads the crawled NDJSON (one taxon per line) and emits one row per species with the four structured lichen traits scraped from each taxon page. Names carry authorship and are resolved against the backbones by the build pipeline.

Usage

```
parse_italic(path)
```

Arguments

path Character. Path to italic.jsonl.

Value

data.frame with canonical_name + lichen trait columns.

parse_iucn	<i>Parse IUCN Red List conservation status from the GBIF Darwin Core Archive</i>
------------	--

Description

Reads the IUCN Red List archive published on GBIF (taxon.txt core plus the Distribution extension that carries threatStatus) and returns one global Red List category per accepted species-rank name. Reading the archive directly avoids the GBIF species/search endpoint, whose threat facet spans every checklist (so a name picks up conflicting categories from regional or erroneous lists) and whose offset ceiling truncates the large categories.

Usage

```
parse_iucn(dir_path)
```

Arguments

dir_path Character. Directory holding the extracted IUCN archive (taxon.txt, distribution.txt).

Value

data.frame with canonical_name + conservation_status.

parse_kew_cvalues	<i>Parse the Kew Plant DNA C-values database (genome size)</i>
-------------------	--

Description

Genome-size estimates for plants, scraped from the paginated Kew C-values search interface as one HTML page per 100 records (the source exposes no bulk export). The data table on each page (Family, Genus, Species, chromosome number, ploidy, 1C DNA amount) is read; the Mean/Min/Max summary-statistic tables are ignored. Records are keyed on the binomial and reduced to one row per taxon: genome size as the 1C DNA amount in picograms, plus chromosome number (2n) and ploidy level where a clean numeric value is given. The 1C DNA amount is reduced by median; the chromosome number and ploidy level are discrete counts whose repeated records are cytotype variants of one species, so they are reduced to the base cytotype by minimum, with _min / _max carrying the range.

Usage

```
parse_kew_cvalues(path)
```

Arguments

path Directory of downloaded C-values HTML pages (or a single page).

Details

Each record names the paper the value was measured in and the method used, so both are kept: original_reference and estimation_method list every distinct value a species' records carry, joined by "; ". The method matters for interpreting the 1C amount, since Feulgen densitometry and flow cytometry are different measurements.

Value

data.frame with canonical_name + genome-size traits.

parse_kew_sid	<i>Parse SER-SID (Kew Seed Information Database) tables to per-species traits</i>
---------------	---

Description

Reduces the per-record SER-SID tables (many measurements per species) to one row per species: seed weight and nutrient contents by median, storage behaviour and fruit type by mode, with a record count for the seed-weight evidence.

Usage

parse_kew_sid(path)

Arguments

path Character. Directory holding the per-table RDS files written by the registry download function (species, seed_weights, storage_behaviour, oil_content, protein_content, morphology).

Value

data.frame with canonical_name and the trait columns thousand_seed_weight (grams per 1000 seeds, median), n_seed_weight_records, storage_behaviour, oil_content_pct, protein_content_pct, lifeform (Raunkiaer code), and fruit_type.

 parse_leda

 Parse LEDA trait files

Description

Each LEDA text dump is a query export: an SQL preamble closed by an `on <date> .` line, then a `;-` separated table whose first line is the header. `.read_leda_table()` reads that structure, and every output column is taken from one named header field of one file, listed in `.leda_trait_spec()`. A named field missing from its file stops the parse rather than falling back to another column.

Usage

```
parse_leda(dir_path)
```

Arguments

`dir_path` Character. Directory containing the LEDA *.txt files.

Details

Numeric traits read LEDA's `single value` field. LEDA fills it for every record as the record's value: the mean where one was reported, otherwise the median, otherwise the reported extreme or the midpoint of the reported range. Records are reduced to the species median, with the gated `<col>_min / <col>_max / <col>_n` spread of `.num_group_spread()`. Categorical traits report the species' most frequent value. Units are LEDA's own (Knevel et al. 2003, *Collecting and measuring standards of life-history traits of the Northwest European flora*): seed mass mg, seed length mm, canopy height m, leaf mass mg, leaf size mm², SLA mm²/mg, LDMC mg/g, terminal velocity m/s, releasing height m.

Two traits need more than a field:

- stem specific density (`ssd.txt`) has no `single value` field, so a record reads its mean, else its median, else the midpoint of its range. The header says g/cm³ and LEDA's validity range is 0-1.5 g/cm³, but most records carry wood densities in kg/m³ (*Quercus robur* 689); a value above 1.5 is read as kg/m³ and divided by 1000.
- floating capacity (`buoyancy.txt`) is the percentage of diaspores still floating after a given time; `floating_capacity_1week_pct` reads the records at LEDA's standard final interval, T6 - 1 week.

Each trait column `<col>` read from a file that records its references has a `<col>_source` column naming the references behind the value: for a categorical trait the records that state the reported value, for a numeric trait every record entering the median. A record's reference is its original reference where LEDA gives one (the publication the value was taken from), with the contributing reference kept as `via`; otherwise its reference. Ids are the first 12 hex digits of the md5 of the citation and `via`, and resolve against the table attached with `attach_references()`. `dispersal_type.txt` is an aggregated query export with no reference field, so `dispersal_type` has no provenance column.

Value

data.frame with canonical_name + LEDA trait columns.

parse_leptraits	<i>Parse LepTraits 1.0 butterfly consensus CSV</i>
-----------------	--

Description

Parse LepTraits 1.0 butterfly consensus CSV

Usage

```
parse_leptraits(path)
```

Arguments

path Character. Path to consensus.csv.

Value

data.frame with canonical_name + butterfly trait columns.

parse_madin	<i>Parse the Madin et al. bacteria/archaea trait database</i>
-------------	---

Description

Reads the species-aggregated condensed_species_NCBI.csv (one row per species). Carries the NCBI taxonomy id as a cross-reference.

Usage

```
parse_madin(path)
```

Arguments

path Character. Path to condensed_species_NCBI.csv.

Value

data.frame with canonical_name + microbial trait columns.

parse_marine_distribution

Parse WoRMS marine distributions rolled up to MEOW ecoregions

Description

Reads the frozen WoRMS distribution snapshot and the MRGID -> MEOW ecoregion crosswalk from `dir_path`, drops records WoRMS flags as doubtful or inaccurate along with those reporting non-presence, expands each remaining distribution record to every MEOW ecoregion its MRGID maps to, and collapses to one row per (species, ecoregion) with a native/introduced status.

Usage

```
parse_marine_distribution(dir_path)
```

Arguments

`dir_path` Character. Directory containing `worms_distributions.jsonl` (or `*.ndjson`) and `mrqid_meow.tsv`.

Value

data.frame with `canonical_name`, `region_code` (MEOW ECO_CODE as a string), `ecoregion`, `province`, `realm`, and `native_status`.

parse_monograph_body_length

Mine body length (mm) from an OCR'd taxonomic monograph

Description

Reads the plain text of one Collembola monograph and returns a body length per species, using the source's layout spec to attach each measurement to a name. The measurement classification is the same core the `plazi_collembola_body_length` enrichment uses. `<prefix>_body_length_mm` is the median of the recorded values, `<prefix>_body_length_min_mm` / `<prefix>_body_length_max_mm` bound them, and `<prefix>_body_length_n` is the number of measurements behind the species' value.

Usage

```
parse_monograph_body_length(path, source)
```

Arguments

path	Character. Path to the monograph text file (an OCR export), or a directory holding one .txt.
source	Character. Which monograph, one of names(.monograph_specs): "stach_1957", "hopkin_2007", "bretfeld_1999".

Value

data.frame with canonical_name and the <prefix>_body_length_* columns, one row per species.

parse_monograph_collembola

Aggregate the frozen monograph body-length extraction to one row per species

Description

Reads the long-format frozen extraction (canonical_name, source, and the body_length_* columns; one row per species per monograph, as [parse_monograph_body_length\(\)](#) produces for each book) and aggregates across the source monographs to one row per species. This is the build reader for the monograph_collembola_body_length enrichment: the source monographs are copyrighted, so the mining runs once and only the extracted values – facts, not the book text – are frozen and redistributed. monograph_body_length_mm is the median of the per-monograph medians, monograph_body_length_min_mm/monograph_body_length_max_mm bound the recorded values, monograph_body_length_n is the total measurement count and monograph_body_length_sources the number of monographs behind the value.

Usage

```
parse_monograph_collembola(path)
```

Arguments

path	Character. Path to the frozen extraction .csv, or a directory containing one .csv.
------	--

Value

data.frame with canonical_name and the monograph_body_length_* columns, one row per species.

parse_ncbi_common_names	<i>Parse NCBI common names from names.dmp</i>
-------------------------	---

Description

Parse NCBI common names from names.dmp

Usage

parse_ncbi_common_names(dir_path)

Arguments

dir_path Character. Directory containing names.dmp.

Value

data.frame with canonical_name + lang (NA) + common_name.

parse_nesttrait	<i>Parse Bird Nest Traits (NestTrait v2)</i>
-----------------	--

Description

Wide table, one row per bird species, with binary presence flags for nest site, nest structure and nest attachment plus mound-builder and brood-parasite flags. The 0/1 indicators are kept, and each of the three flag groups is also collapsed to a single pipe-delimited categorical column (nest_structure, nest_site, nest_attachment) so the taxify trait verb can consume them; multi-modal species carry every set modality (e.g. "tree|nontree|cliff_bank").

Usage

parse_nesttrait(path)

Arguments

path Path to NestTrait_v2.csv.

Value

data.frame with canonical_name + nest trait flags + the three collapsed categorical columns.

 parse_noddb

Parse NodDB, the global database of root-symbiotic nitrogen fixation

Description

NodDB database v1.3b.xlsx from the PlutoF deposit behind Tedersoo et al. (2018, Journal of Vegetation Science 29:560-568). Nodulation is recorded per plant genus, not per species, so this is a genus-keyed source in the same shape as FungalRoot: canonical_name carries the genus and taxify joins it on genus.

Usage

```
parse_noddb(path)
```

Arguments

path Character. Path to the downloaded .xlsx.

Details

The sheet puts its citation on row 1 and the header on row 2, so the header is read positionally rather than by read.csv's first-row assumption.

Consensus estimate is the authors' verdict and is kept verbatim. The canonical nodulation_type collapses it to the symbiont actually involved. The collapse is an exact lookup rather than a regex because the vocabulary contains Rhizobia, likely_Rhizobia and unlikely_Rhizobia, each of which contains the one before it – a pattern match in the wrong order would read every negative verdict as a positive one. unlikely_* is the authors' judgement that the genus does not nodulate and reads as none.

Value

data.frame with canonical_name (genus) plus nodulation columns.

 parse_nztd

Parse NZTD New Zealand marine benthos traits

Description

Two-row header (trait category + modality) with fuzzy affinity scores; each trait category is reduced to its dominant modality (with the source labels), one row per species.

Usage

```
parse_nztd(path)
```

Arguments

path Path to the NZTD xlsx.

Value

data.frame with canonical_name + benthic traits.

parse_octocoral	<i>Parse Octocoral Trait Database (v2.2)</i>
-----------------	--

Description

Long-format trait table; reduced to one row per species. Geographic and bioregion fields are dropped; biological traits (colony morphology, polyp, skeleton, symbiosis, feeding) are kept (numeric by median, categorical by mode).

Usage

parse_octocoral(path)

Arguments

path Path to OctocoralTraits_v2_2.csv.

Value

data.frame with canonical_name + traits.

parse_odonata	<i>Parse Odonate Phenotypic Database (categorical traits)</i>
---------------	---

Description

Multi-row table (several records per species); reduced to one row per species (mode) for behavioural and ecological categorical traits. Multi-valued cells are kept verbatim.

Usage

parse_odonata(path)

Arguments

path Path to the OPD CSV.

Value

data.frame with canonical_name + odonate traits.

`parse_ott_common_names`*Parse OTT common names from synonyms.tsv + taxonomy.tsv*

Description

Parse OTT common names from synonyms.tsv + taxonomy.tsv

Usage

```
parse_ott_common_names(dir_path)
```

Arguments

`dir_path` Character. Directory containing synonyms.tsv + taxonomy.tsv.

Value

data.frame with canonical_name + lang (NA) + common_name.

`parse_pantheria`*Parse PanTHERIA mammal life-history traits (TSV)*

Description

Parse PanTHERIA mammal life-history traits (TSV)

Usage

```
parse_pantheria(path)
```

Arguments

`path` Character. Path to PanTHERIA.txt.

Value

data.frame with canonical_name + life-history columns.

<code>parse_parravicini</code>	<i>Parse Parravicini reef-fish trophic guilds</i>
--------------------------------	---

Description

Each species carries one trophic-guild code per contributing expert; the cross-expert consensus (mode) is taken as the species trophic guild.

Usage

`parse_parravicini(path)`

Arguments

`path` Path to converted_experts_classification.csv.

Value

data.frame with canonical_name + trophic_guild.

<code>parse_pelagic</code>	<i>Parse Pelagic Species Trait Database</i>
----------------------------	---

Description

Wide table, one row per pelagic species (fish, cephalopods, gelatinous), of depth/temperature envelope, size, habitat, defence and gregariousness traits.

Usage

`parse_pelagic(path)`

Arguments

`path` Path to the pelagic trait CSV.

Value

data.frame with canonical_name + traits.

parse_phylacine	<i>Parse PHYLACINE v1.2 trait data</i>
-----------------	--

Description

One row per mammal species (including recently and prehistorically extinct). The binomial is stored with an underscore (Genus_species) and is converted to a space-separated canonical name.

Usage

```
parse_phylacine(path)
```

Arguments

path	Character. Path to Trait_data.csv.
------	------------------------------------

Details

Mass.g is partly modelled: PHYLACINE gap-fills data-poor and extinct species. The source Mass.Method flag is kept verbatim as mass_method, alongside a coarse mass_method_class (reported / estimated / imputed) so a modelled mass is never served as a measurement where PHYLACINE is the sole source for a species.

Value

data.frame with canonical_name + mammal traits.

parse_plantatt	<i>Parse PLANTATT (attributes of British and Irish plants)</i>
----------------	--

Description

One row per vascular-plant taxon. Retained traits are the five Ellenberg indicator values (light, moisture, reaction, nitrogen, salt), maximum height (cm), and the source's life-form, woodiness and native-status codes (kept verbatim).

Usage

```
parse_plantatt(path)
```

Arguments

path	Path to PLANTATT_19_Nov_08.xls (or its directory).
------	--

Value

data.frame with canonical_name + British/Irish plant attributes.

```
parse_plazi_collembola_body_length
```

Parse Plazi Collembola treatments into per-species body length

Description

Reads the Darwin Core archives harvested by `harvest_plazi_dwca()`, keeps the Collembola taxa, and extracts body length (mm) from the prose of their treatments. `plazi_body_length_mm` is the median of the recorded values, `plazi_body_length_min_mm` / `plazi_body_length_max_mm` bound them, and `plazi_body_length_n` / `plazi_body_length_sources` are the number of measurements and of distinct treatment papers behind the species' value.

Usage

```
parse_plazi_collembola_body_length(
  path,
  sections = c("description", "diagnosis", "type_taxon", "discussion", "biology_ecology",
    "materials_examined", "distribution", "etymology")
)
```

Arguments

<code>path</code>	Character. Directory holding the harvested .zip archives.
<code>sections</code>	Character. Treatment section types to read. The default set is every section that was measured to carry body lengths; narrowing it to <code>c("description", "diagnosis")</code> raises precision slightly at the cost of about 6 percent of the species.

Value

data.frame with `canonical_name` and the `plazi_body_length_*` columns, one row per species.

```
parse_pottier
```

Parse Pottier amphibian heat tolerance

Description

Per-measurement upper thermal-limit records reduced to species medians.

Usage

```
parse_pottier(path)
```

Arguments

<code>path</code>	Path to <code>Curated_data.csv</code> .
-------------------	---

Value

data.frame with canonical_name + thermal/size traits.

parse_quimbayo	<i>Parse Quimbayo reef-fish traits</i>
----------------	--

Description

Wide table, one row per reef-fish species; a curated set of size, ecology, depth, trophic and behavioural traits is kept.

Usage

```
parse_quimbayo(path)
```

Arguments

path Path to the Fish_aspects CSV.

Value

data.frame with canonical_name + reef-fish traits.

parse_ramond	<i>Parse the Ramond et al. plankton protist trait database</i>
--------------	--

Description

Morphological, behavioural and ecological traits of marine protists, keyed on the finest determined name (Last), which is genus-level or higher. The three source tables share the same taxa with overlapping trait columns; they are stacked and reduced to one row per genus (numeric traits – cell size, in micrometres – by median, categorical traits by mode). Undetermined entries and keys that are not a taxon name are dropped; a higher-rank name is kept but never matches the genus of a taxify result.

Usage

```
parse_ramond(path)
```

Arguments

path Directory holding the three source CSVs.

Value

data.frame with canonical_name (genus) + protist traits.

parse_repttraits	<i>Parse ReptTraits (Etard et al. 2024) reptile traits (XLSX from Figshare)</i>
------------------	---

Description

ReptTraits is the global ecological-trait dataset for reptiles, built on the Reptile Database taxonomy. The header names are mapped explicitly (not by fuzzy pattern) so the distribution/environment block and the morphology block resolve to the correct source columns. Columns surface a per-species range signal (biogeographic realm, elevation, climate) plus body-size and life-history traits across all reptiles (not lizards only).

Usage

parse_repttraits(path)

Arguments

path Character. Path to the ReptTraits XLSX (or CSV/TSV).

Value

data.frame with canonical_name + reptile trait columns.

parse_rimet_phyto	<i>Parse Rimet & Druart phytoplankton metrics</i>
-------------------	---

Description

Wide table, one row per taxon, of cell and colony morphometrics. Cell-level size/volume/area metrics are kept.

Usage

parse_rimet_phyto(path)

Arguments

path Path to the phytoplankton metrics xlsx.

Value

data.frame with canonical_name + cell metrics.

parse_saproxylic	<i>Parse Hagge saproxylic beetle morphology</i>
------------------	---

Description

Wide table, one row per deadwood-beetle species (underscore names), of body and appendage morphometrics.

Usage

```
parse_saproxylic(path)
```

Arguments

path	Path to the trait CSV.
------	------------------------

Value

data.frame with canonical_name + morphometrics.

parse_sealifebase	<i>Parse SeaLifeBase species traits (via rfishbase)</i>
-------------------	---

Description

Non-fish companion to [parse_fishbase\(\)](#) (molluscs, crustaceans, echinoderms, marine mammals, reptiles, etc.). Shares the same trait columns.

Usage

```
parse_sealifebase(path)
```

Arguments

path	Ignored; rfishbase fetches data directly.
------	---

Value

data.frame keyed on canonical_name with eight trait columns.

<code>parse_sharkipedia</code>	<i>Parse Sharkipedia elasmobranch life-history traits</i>
--------------------------------	---

Description

Long-format trait table (one row per observation); reduced to one row per species (numeric traits by median) for a curated set of growth, length, age and reproduction traits.

Usage

```
parse_sharkipedia(path)
```

Arguments

`path` Path to the Sharkipedia traits CSV.

Value

data.frame with canonical_name + traits.

<code>parse_sheld</code>	<i>Parse SHELD US freshwater mussel traits</i>
--------------------------	--

Description

Wide species trait matrix, one row per mussel species.

Usage

```
parse_sheld(path)
```

Arguments

`path` Path to the SHELD species trait matrix xlsx.

Value

data.frame with canonical_name + mussel traits.

parse_spider_traits	<i>Parse the World Spider Trait Database CSV export</i>
---------------------	---

Description

The full export is long format with ~36% access-restricted rows (value = "access restricted") that are dropped. A set of high-coverage morphometric and ecological traits is pivoted to one row per species.

Usage

```
parse_spider_traits(path)
```

Arguments

path Character. Path to the WST CSV export.

Value

data.frame with canonical_name + spider trait columns.

parse_sworm	<i>Parse the sWorm global earthworm dataset into per-species ecological groups</i>
-------------	--

Description

SppOccData_sWorm_<date>.csv from the sWorm data release (Phillips et al. 2021, Scientific Data 8:136), the sDiv-funded compilation behind the global earthworm diversity synthesis.

Usage

```
parse_sworm(path)
```

Arguments

path Character. Directory holding the unpacked sWorm release.

Details

The release is a community dataset – species occurrences with abundance and biomass at 10,840 sites – and one column of it is a species trait: the Bouche ecological group each earthworm was assigned from its feeding and burrowing behaviour. That column is constant within a species across the whole compilation (171 of 172 species carry exactly one group and none carries two), so collapsing it to one row per species loses nothing.

Abundance and wet biomass are deliberately left out. Both ship in mixed units in the same column – individuals, individuals per m2 and per m3; grams, g/m2 and mg/m2 – mixing a per-individual measure with a per-area density, so neither has a unit that survives aggregation to the species. The native/non-native column is left out for a different reason: it is a status at a site rather than a property of the species, and a species is native in one place and introduced in another.

Earthworms are otherwise absent from the bundled trait sources.

Value

data.frame with canonical_name and ecological_group.

parse_tetradensity	<i>Parse TetraDENSITY population density</i>
--------------------	--

Description

Per-record population densities reduced to species medians. Only the dominant ind/km2 unit is kept (mixing with pairs/km2 or ind/ha would be incorrect).

Usage

parse_tetradensity(path)

Arguments

path Path to TetraDENSITY CSV.

Value

data.frame with canonical_name + density_ind_km2.

parse_thermofresh	<i>Parse the Freshwater thermal-tolerance database (ThermoFresh)</i>
-------------------	--

Description

Per-test critical thermal limits (Bayat et al.) for freshwater fish, invertebrates and amphibians. Each source row is one tolerance test; the processed tests are joined to the taxonomy table on tax_id, restricted to species-level taxa, and reduced to one row per species (median per metric). The retained metrics are the critical thermal maximum (ctmax), minimum (ctmin), the median lethal temperature (lt50) and the lethal maximum / minimum (ltmax, ltmin); all values are in degrees Celsius.

Usage

parse_thermofresh(path)

Arguments

path Directory holding the extracted archive (with a data/ subdirectory) or a path directly to that data/ directory.

Details

The archive ships the peer-reviewed tables as data/*_final.csv and keeps the pre-review submission beside them under data/initial_submission/, so the readers below match the _final names exactly.

Value

data.frame with canonical_name + freshwater thermal-limit traits.

parse_tree_of_sex	<i>Parse the Tree of Sex database (plants + vertebrates + invertebrates)</i>
-------------------	--

Description

Row-binds the three sub-tables on a common core keyed on Genus species, adding a taxon_group discriminator. Group-specific traits (selfing for plants, environmental sex determination for vertebrates, haplodiploidy for invertebrates) are kept and are NA for the other groups.

Usage

```
parse_tree_of_sex(path)
```

Arguments

path Character. Directory holding the three Tree of Sex CSVs.

Value

data.frame with canonical_name + sex-determination traits.

parse_usda_fungus_host	<i>Parse USDA Fungus-Host records into per-fungus host breadth</i>
------------------------	--

Description

Rolls up the USDA National Fungus Collections fungus->host table to one row per accepted fungus name with the number of distinct host binomials and distinct host genera.

Usage

parse_usda_fungus_host(path)

Arguments

path Character. Path to the Fungus-Host CSV.

Value

data.frame with canonical_name, fungus_host_count, fungus_host_genus_count.

parse_useful_plants	<i>Parse the World Checklist of Useful Plant Species PDF</i>
---------------------	--

Description

The checklist is published only as a typeset PDF. Each species is a name line followed by a data line <IPNI LSID> | <space-separated use codes> | [CWR] | [sources]. The ten Level-1 use codes are exploded into boolean columns and a crop-wild-relative flag is set from the optional CWR token.

Usage

parse_useful_plants(path)

Arguments

path Character. Path to the WCUP PDF.

Value

data.frame with canonical_name + use-category booleans.

parse_virion	<i>Parse VIRION into per-host virus association breadth</i>
--------------	---

Description

virion.csv.gz plus tax_table.csv.gz from the VIRION data package (Carlson et al. 2022, mBio 13:e0298521), the merged host-virus association network. The association table keys hosts and viruses by hash, so the taxonomy table supplies the names.

Usage

```
parse_virion(path)
```

Arguments

path	Character. Directory holding the downloaded VIRION files.
------	---

Details

Only per-host aggregates are written: the number of distinct viruses recorded from the host, the number of distinct virus families, and the number of association records behind those counts. The association records themselves are not redistributed, the same treatment GloBI and InvaCost get.

Counts are association breadth as recorded, not a biological property of the host: sampling effort dominates, which is why *Homo sapiens* leads with 936 distinct viruses. The record count travels alongside so the effort behind a number is visible.

Value

data.frame with canonical_name plus virus association counts.

parse_wcvp	<i>Parse WCVP names + distribution (from extracted ZIP directory)</i>
------------	---

Description

Parse WCVP names + distribution (from extracted ZIP directory)

Usage

```
parse_wcvp(dir_path)
```

Arguments

dir_path	Character. Directory containing wcvp_names + wcvp_distribution.
----------	---

Value

data.frame with canonical_name + tdwg_code + native_status.

<code>parse_zanne</code>	<i>Parse Zanne et al. 2014 woodiness CSV</i>
--------------------------	--

Description

Parse Zanne et al. 2014 woodiness CSV

Usage

`parse_zanne(path)`

Arguments

`path` Character. Path to GlobalWoodinessDatabase.csv.

Value

data.frame with canonical_name + woodiness.

<code>parse_zooplankton</code>	<i>Parse Global Zooplankton Trait Database</i>
--------------------------------	--

Description

Long-format trait records reduced to one row per species (numeric by median, categorical by mode). Binary one-hot decompositions of the categorical traits are not carried (the categorical parent is kept instead).

Usage

`parse_zooplankton(path)`

Arguments

`path` Path to the level-2 trait CSV.

Value

data.frame with canonical_name + zooplankton traits.

precompute_backbone	<i>Full precompute pipeline: keys + synonym embedding</i>
---------------------	---

Description

Adds the matching-key columns and embedded synonym info that taxify’s runtime expects. Operates on a normalized backbone (column names from [normalize_backbone\(\)](#)).

Usage

```
precompute_backbone(df, synonym_pattern = "SYNONYM")
```

Arguments

- df A normalized backbone data.frame.
- synonym_pattern Regex for synonym detection.

Value

The data.frame ready for build_vtr().

probe_upstream_identity	<i>Ask a source host what identity its newest version carries</i>
-------------------------	---

Description

The single entry point to the probes: a build calls it to record what it read, and the weekly check calls it to see what upstream now offers. Both therefore speak in the same identity, which is what makes the two comparable.

Usage

```
probe_upstream_identity(source_url, record = NULL)
```

Arguments

- source_url Character. The URL a build downloads its source from.
- record Character or NULL. The source’s host record (registry source_record, else source_doi); see [check_figshare_version\(\)](#). A source downloading from several URLs (url ; url) is probed part by part, and its identity is theirs joined: it moves when any part moves, it is pinned only when every part is, and it dates from its newest part.

Value

The probe's result (id, version, date, optionally pinned, url), or NULL when no probe covers the host or the host cannot be reached (for a multi-URL source: any of its hosts).

publish_enrichment_release

Publish enrichment .vtr files to the rolling enrichment release

Description

Every enrichment is published under one shared, rolling release tag (enrichment-<version>), unlike backbones which get a tag each. The release is therefore created only when missing and never deleted: assets are uploaded with --clobber, which replaces only the same-named file and leaves every other enrichment's .vtr in place. Deleting and recreating the tag (as the per-backbone workflow does) would wipe all the other enrichments' assets.

Usage

```
publish_enrichment_release(
  version,
  vtr_paths,
  repo = "gcol33/taxifydb",
  notes = NULL
)
```

Arguments

version	Character. Rolling release version (the enrichment-<version> tag).
vtr_paths	Character vector. Paths to the enrichment .vtr files to upload. Basenames become the release asset names.
repo	Character. GitHub repo (e.g. "gcol33/taxifydb").
notes	Character or NULL. Release notes, used only when the release is created for the first time.

Value

The release tag (invisibly).

publish_release	<i>Create a GitHub release and upload backbone artifacts</i>
-----------------	--

Description

Uses the gh CLI. Assumes gh is authenticated and on PATH.

Usage

```
publish_release(
  backend_name,
  version,
  vtr_path,
  delta_path = NULL,
  meta_path = NULL,
  extras = character(0L),
  repo = "gcol33/taxifydb",
  notes = NULL
)
```

Arguments

backend_name	Character. Backend identifier.
version	Character. The source release the build recorded in its .meta; a different value is an error.
vtr_path	Character. Path to the main .vtr file.
delta_path	Character or NULL. Path to the .xdelta file.
meta_path	Character or NULL. Path to the .meta sidecar.
extras	Character vector. Paths to additional sidecar artifacts (e.g., col_species_profile.vtr) that should be uploaded with the release and recorded in the manifest. Must exist on disk; basenames are used as the manifest entry names.
repo	Character. GitHub repo (e.g., "gcol33/taxifydb").
notes	Character. Release notes.

Details

The release is created when missing and never deleted. Publishing again under an existing tag replaces the rolling assets in place (`--clobber`) and adds the new build's content-addressed copy beside those of earlier cuts, so every `content_url` a manifest has recorded for the tag keeps resolving. A rolling .vtr published before those copies existed is preserved under its own content-addressed name before it is replaced.

Value

The release tag (invisibly).

read_algaebase	<i>Read and normalize a fetched list of AlgaeBase records</i>
----------------	---

Description

Read and normalize a fetched list of AlgaeBase records

Usage

```
read_algaebase(records, verbose = TRUE)
```

Arguments

records	A list of raw search records (from <code>download_algaebase()</code>).
verbose	Logical.

Value

A normalized data.frame.

read_avilist	<i>Read and normalize the AviList checklist</i>
--------------	---

Description

Read and normalize the AviList checklist

Usage

```
read_avilist(xlsx_path, verbose = TRUE)
```

Arguments

xlsx_path	Character. Path to the AviList extended workbook.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_col	<i>Read and normalize the COL Taxon.tsv</i>
----------	---

Description

Read and normalize the COL Taxon.tsv

Usage

```
read_col(col_dir, verbose = TRUE)
```

Arguments

col_dir	Character. Path to the extracted COL directory.
verbose	Logical.

Value

A normalized data.frame.

read_colxr	<i>Read and normalize the COL XR export</i>
------------	---

Description

Read and normalize the COL XR export

Usage

```
read_colxr(tsv_path, verbose = TRUE)
```

Arguments

tsv_path	Character. Path to the extracted COL XR TSV.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_dmp	<i>Read a pipe-delimited NCBI .dmp file</i>
----------	---

Description

Read a pipe-delimited NCBI .dmp file

Usage

```
read_dmp(path, col_names)
```

Arguments

path	Character. Path to the .dmp file.
col_names	Character vector of column names.

Value

A data.frame.

read_euromed	<i>Read and normalize the Euro+Med CDM snapshot</i>
--------------	---

Description

Read and normalize the Euro+Med CDM snapshot

Usage

```
read_euromed(jsonl_path, nodes_path, verbose = TRUE)
```

Arguments

jsonl_path	Character. Path to euromed.jsonl (one accepted taxon per line, synonyms nested).
nodes_path	Character. Path to nodes.tsv (genus/suprageneric treeIndex paths for family resolution).
verbose	Logical.

Value

A normalized data.frame in the unified backbone schema.

read_evc	<i>Read and normalize the EuroVegChecklist into the syntaxon backbone schema</i>
----------	--

Description

One row per accepted unit (class, order, alliance) and one per synonym. A synonym is every name the unit's Nomenclature page lists, plus the name the original EuroVegChecklist (Mucina et al. 2016) used where the current version renamed the unit. Every row carries the lineage of its accepted unit in class / order / alliance, so a match at any rank rolls up.

Usage

```
read_evc(synonyms_path, units_path, synonyms_2016_path = NULL, verbose = TRUE)
```

Arguments

synonyms_path	Path to evc_synonyms.jsonl.
units_path	Path to the unit list .xlsx.
synonyms_2016_path	Path to evc2016_synonyms.tsv, or NULL. Read only for units whose Nomenclature page was truncated.
verbose	Logical.

Value

A data.frame: taxon_id, canonical_name, authorship, taxon_rank, taxonomic_status, accepted_name_usage_id, evc_code, nomenclatural_note, remark, class, order, alliance, association (always NA: the EVC stops at alliances), evc_mnemonic, description.

read_fungorum	<i>Read and normalize the Species Fungorum Plus ColDP files</i>
---------------	---

Description

Read and normalize the Species Fungorum Plus ColDP files

Usage

```
read_fungorum(fungorum_dir, verbose = TRUE)
```

Arguments

fungorum_dir	Character. Directory containing Taxon.tsv, Name.tsv, Synonym.tsv.
verbose	Logical.

Value

A normalized data.frame.

read_gbif	<i>Read and normalize the GBIF backbone</i>
-----------	---

Description

Read and normalize the GBIF backbone

Usage

```
read_gbif(gz_path, counts = NULL, verbose = TRUE)
```

Arguments

- | | |
|---------|--|
| gz_path | Character. Path to simple.txt.gz. |
| counts | Named numeric vector of occurrence counts by taxon key, from read_gbif_occurrence_counts() , or NULL for none. |
| verbose | Logical. |

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_gbif_occurrence_counts	<i>Read a GBIF occurrence-count snapshot</i>
-----------------------------	--

Description

Read a GBIF occurrence-count snapshot

Usage

```
read_gbif_occurrence_counts(path)
```

Arguments

- | | |
|------|---------------------------------|
| path | Character. Path to the .tsv.gz. |
|------|---------------------------------|

Value

Named numeric vector of counts by taxon key.

read_itis	<i>Read and normalize the ITIS SQLite database</i>
-----------	--

Description

Read and normalize the ITIS SQLite database

Usage

```
read_itis(sqlite_path, verbose = TRUE)
```

Arguments

sqlite_path	Character. Path to the ITIS .sqlite file.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_lcvp	<i>Read and normalize the LCV reference table</i>
-----------	---

Description

Read and normalize the LCV reference table

Usage

```
read_lcvp(rda_path, verbose = TRUE)
```

Arguments

rda_path	Character. Path to tab_lcvp.rda.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_lpsn	<i>Read and normalize the LPSN checklist</i>
-----------	--

Description

Read and normalize the LPSN checklist

Usage

```
read_lpsn(lpsn_dir, verbose = TRUE)
```

Arguments

lpsn_dir	Character. Directory holding NameUsage.tsv.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_mdd	<i>Read and normalize the Mammal Diversity Database</i>
----------	---

Description

Read and normalize the Mammal Diversity Database

Usage

```
read_mdd(dir, verbose = TRUE)
```

Arguments

dir	Character. Directory holding the extracted MDD CSVs.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_meow	<i>Parse the MEOW ecoregion GeoJSON into a long-format vertex table</i>
-----------	---

Description

One row per polygon vertex. `code` is the MEOW ecoregion code (`ECO_CODE`), `geom` indexes the polygon within a feature (a `MultiPolygon` has several), `ring` indexes the ring within a polygon (1 is the outer boundary, 2+ are holes), and `seq` is the vertex order within a ring. This captures the full polygon topology so the runtime can rebuild native, terra, or sf geometry from the same source, exactly like `read_wgsrpd()`.

Usage

```
read_meow(path)
```

Arguments

<code>path</code>	Character. Path to a MEOW ecoregion GeoJSON file.
-------------------	---

Details

The ecoregion code property is matched case-insensitively, preferring `ECO_CODE`; falling back to `ECO_CODE_X`, then any `eco*code` property.

Value

A data.frame with columns `code`, `geom`, `ring`, `seq`, `lon`, `lat`, sorted by `code`, `geom`, `ring`, `seq`.

read_ncbi	<i>Read and normalize the NCBI taxonomy dump</i>
-----------	--

Description

Read and normalize the NCBI taxonomy dump

Usage

```
read_ncbi(dump_dir, verbose = TRUE)
```

Arguments

<code>dump_dir</code>	Character. Path to the extracted taxdump directory.
<code>verbose</code>	Logical.

Value

A normalized data.frame ready for `precompute_backbone()`.

read_ott	<i>Read and normalize the OTT taxonomy</i>
----------	--

Description

Read and normalize the OTT taxonomy

Usage

```
read_ott(ott_dir, verbose = TRUE)
```

Arguments

ott_dir	Character. Path to the extracted OTT directory.
verbose	Logical.

Value

A normalized data.frame.

read_reptiledb	<i>Read and normalize the Reptile Database</i>
----------------	--

Description

Read and normalize the Reptile Database

Usage

```
read_reptiledb(paths, verbose = TRUE)
```

Arguments

paths	Named list from download_reptiledb() .
verbose	Logical.

Value

A normalized backbone data.frame.

read_vegcz	<i>Read the Czech vegetation classification into the syntaxon backbone schema</i>
------------	---

Description

One row per accepted unit (class, alliance, association) and one per name its Pladias Nomenclature page lists as a synonym. A name cited in another author's sense (sensu ... non ..., a pseudonym) is MISAPPLIED. Every row carries the lineage of its accepted unit in class / alliance / association (order is NA: the national classification does not use orders).

Usage

```
read_vegcz(synonyms_path, units_path, verbose = TRUE)
```

Arguments

synonyms_path	Path to vegcz_synonyms.jsonl.
units_path	Path to the unit list .xlsx.
verbose	Logical.

Value

A data.frame in the shape of [read_evc\(\)](#), with vegcz_code, description_cz and description_en for its own columns.

read_wcvp	<i>Read and normalize the WCV names file</i>
-----------	--

Description

Read and normalize the WCV names file

Usage

```
read_wcvp(names_path, verbose = TRUE)
```

Arguments

names_path	Character. Path to wcvp_names.csv.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_wfo	<i>Read and normalize the WFO classification file</i>
----------	---

Description

Read and normalize the WFO classification file

Usage

```
read_wfo(txt_path, verbose = TRUE)
```

Arguments

txt_path	Character. Path to the WFO classification.txt file.
verbose	Logical.

Value

A normalized data.frame ready for [precompute_backbone\(\)](#).

read_wgsrpd	<i>Parse the WGSRPD Level 3 GeoJSON into a long-format vertex table</i>
-------------	---

Description

One row per polygon vertex. geom indexes the polygon within a feature (a MultiPolygon has several), ring indexes the ring within a polygon (1 is the outer boundary, 2+ are holes), and seq is the vertex order within a ring. This captures the full polygon topology so the runtime can rebuild native, terra, or sf geometry from the same source.

Usage

```
read_wgsrpd(path)
```

Arguments

path	Character. Path to a WGSRPD Level 3 GeoJSON file.
------	---

Value

A data.frame with columns code, geom, ring, seq, lon, lat, sorted by code, geom, ring, seq.

read_worms	<i>Read and normalize the WoRMS taxonomy</i>
------------	--

Description

Read and normalize the WoRMS taxonomy

Usage

```
read_worms(worms_dir, verbose = TRUE)
```

Arguments

worms_dir	Character. Path to the extracted DwC-A directory.
verbose	Logical.

Value

A normalized data.frame.

register_backbones	<i>Names of the backbones the genus register and backend coverage union</i>
--------------------	---

Description

Names of the backbones the genus register and backend coverage union

Usage

```
register_backbones()
```

Value

Character vector of backend identifiers.

resolve_betsi_codes	<i>Resolve BETSI species codes to binomials</i>
---------------------	---

Description

The code-keyed BETSI-derived matrices (INRAE UU2FQT / UCYSLH, Bonfanti 2022) key traits on a TOKEN_SPP code with no published legend, where SPP is the first three letters of the specific epithet and TOKEN abbreviates the genus. This decodes each code against a reference pool of binomials: the token selects one or more pool genera and the epithet prefix selects the species within them. A single pool species resolves directly; a collision is split by body length when both the code and its candidates carry one; a genus-level _X code, an unmapped token, or a code with no pool candidate is left unresolved for the caller to tag and drop rather than guess (label, never decide).

Usage

```
resolve_betsi_codes(
  codes,
  pool,
  code_bl = NULL,
  pool_bl = NULL,
  genus_dict = NULL
)
```

Arguments

codes	Character vector of TOKEN_SPP codes.
pool	Character vector of candidate binomials.
code_bl	Named numeric mapping code -> estimated body length (mm), or NULL. Only used to split a collision.
pool_bl	Named numeric mapping binomial -> body length (mm), or NULL.
genus_dict	Named character vector (or list) mapping a genus token to its genus name(s), or NULL for the plain first-three-letters rule.

Details

The token is resolved two ways. Bonfanti 2022 uses the plain first three letters of the genus (ISO_PAL for *Isotomurus palustris*), and with genus_dict = NULL a three-letter token matches every pool genus sharing those three letters. INRAE instead disambiguates colliding genera with bespoke abbreviations – the Iso* genera split into ISO = *Isotomiella*, ISA = *Isotoma*, ISU = *Isotomurus*, ISODES = *Isotomodes* – so pass that mapping as genus_dict and a listed token resolves to exactly its genus, with the three-letter rule kept only as the fallback for unlisted three-letter tokens. A token that is neither listed nor a plain three-letter prefix is left unresolved rather than forced onto a near-match; INRAE’s verified mapping is shipped in inst/extdata/betsi/inrae_genus_abbr.csv.

The reference pool is supplied by the caller; in practice it is the union of accepted binomials across taxifydb’s built Collembola assets (betsi_collembola_body_length, ellers_collembola, betsi_collembola_traits, the monograph and Plazi length sources), with pool_bl their body lengths.

Value

A data.frame with one row per input code: code, binomial (NA when unresolved) and method (why it resolved or did not).

See Also

[gen_spe\(\)](#)

resolve_enrichment_names

Resolve enrichment names against all taxify backends

Description

Takes an enrichment data.frame with canonical_name + trait columns and expands it so that each source name maps to all unique accepted_name values across the requested backends.

Usage

```
resolve_enrichment_names(  
  df,  
  group_cols = NULL,  
  backends = list_backends(),  
  verbose = TRUE,  
  use_lookup = TRUE,  
  reverse_hop = TRUE,  
  strict = FALSE,  
  reduce_fn = NULL,  
  grain = c("species", "genus"),  
  kingdom = NULL  
)
```

Arguments

df	A data.frame with at least a canonical_name column.
group_cols	Character vector of grouping columns. Deduplication uses canonical_name + group_cols as the key. Default NULL.
backends	Character vector of backend names. Default: every taxify backbone (list_backends()).
verbose	Logical.
use_lookup	Logical. Try the hash-join fast path first. Default TRUE.
reverse_hop	Logical. Add the reverse hop; see resolve_name_map() .
strict	Logical. Error, rather than warn, when a requested backbone has no name_lookup.vtr. Set for production asset builds.

reduce_fn	Function (df, group_cols) collapsing the expanded table to one row per accepted name (plus group_cols). Defaults to keeping the trait-richest row (.dedup_keep_richest()); an enrichment whose value is a reduction over records (e.g. an earliest first-record year) supplies its own so synonyms collapse by that rule, not by row richness.
grain	"species" (default) or "genus"; see resolve_name_map() .
kingdom	Character vector or NULL; the kingdoms the source covers. See resolve_name_map() .

Details

By default tries the fast hash-join path against per-backend name_lookup.vtr files in the user's taxify data directory (built by [build_all_name_lookups\(\)](#)); falls back to per-name-per-backend [taxify::taxify\(\)](#) if no lookup files are found.

Value

The expanded data.frame, carrying a resolved_backbones attribute naming the backbones the expansion actually reached.

resolve_hierarchy	<i>Resolve family/genus by walking a parent-child hierarchy</i>
-------------------	---

Description

Many backends (ITIS, NCBI, OTT) store taxonomy as a parent-child tree without explicit family/genus columns. This function walks up the tree to find the nearest ancestor at each target rank.

Usage

```
resolve_hierarchy(df, target_ranks = c("family", "genus"), max_depth = 20L)
```

Arguments

df	A data.frame with columns id, parent_id, rank, name.
target_ranks	Character vector of ranks to resolve (e.g., c("family", "genus")).
max_depth	Maximum number of hops (default 20).

Value

The input data.frame with new columns named after target_ranks, containing the resolved ancestor name at that rank.

resolve_name_map	<i>Map names to their cross-backbone accepted names</i>
------------------	---

Description

Resolves a vector of source names to every distinct accepted name they map to across the requested taxify backends, returning the long (input_name, accepted_name) mapping used by [resolve_enrichment_names\(\)](#). Names that resolve nowhere are self-mapped (accepted_name == input_name).

Usage

```
resolve_name_map(
  names,
  backends = list_backends(),
  verbose = TRUE,
  use_lookup = TRUE,
  reverse_hop = TRUE,
  strict = FALSE,
  grain = c("species", "genus"),
  kingdom = NULL
)
```

Arguments

names	Character vector of source names.
backends	Character vector of backend names. Default: every taxify backbone (list_backends()).
verbose	Logical.
use_lookup	Logical. Try the hash-join fast path first. Default TRUE.
reverse_hop	Logical. Add the reverse hop that reaches an accepted name only one backbone keeps. FALSE is the forward image alone, the behaviour before this was fixed.
strict	Logical. Error, rather than warn, when a requested backbone has no name_lookup.vtr. Set for production asset builds.
grain	"species" (default) or "genus". A genus-grain source is joined on the genus of a taxify result, so only accepted names that are themselves a genus name are kept: a backbone that files a source genus as a subgenus (Camponotus (Forelophilus)) or resolves it onto a species gives a key no genus can match, and one that would carry genus-level traits onto a single species. A source name left with no genus-shaped accepted name maps to itself, as an unresolved name does.
kingdom	Character vector or NULL. The kingdoms the source covers (any spelling taxify::normalize_kingdom_g folds). When given, a mapping is kept only if some backbone that supplies it places the accepted name inside that set or records no kingdom for it; the single-kingdom backbones that carry no kingdom column count as their fixed kingdom (taxify::backbone_fixed_kingdom()). This replaces the consensus vote,

which cannot protect a genus-grain source: genus names are homonyms across the zoological and botanical codes far more often than binomials are. Applies to the lookup fast path only.

Details

Exposed so that rollup enrichments (host breadth, interaction degree) can aggregate at the accepted-name grain – unioning records across synonyms before counting – instead of pre-aggregating per source name and losing counts when several synonyms later collapse onto one accepted name.

Value

data.frame with columns input_name, accepted_name.

rfishbase_release	<i>Latest rfishbase data release for a server</i>
-------------------	---

Description

rfishbase serves FishBase and SeaLifeBase as dated snapshots named YY.MM ("26.06" is June 2026). The newest is resolved once per build and passed to every table read, so the taxa and synonyms come from the snapshot the version names.

Usage

```
rfishbase_release(server = c("fishbase", "sealifebase"))
```

Arguments

server Either "fishbase" or "sealifebase".

Value

A list with release (rfishbase's YY.MM label), version (YYYY.MM) and date (YYYY-MM, the snapshot's month).

sha256	<i>Compute SHA-256 checksum of a file</i>
--------	---

Description

Compute SHA-256 checksum of a file

Usage

sha256(path)

Arguments

path Character. Path to the file.

Value

Character. Hex-encoded SHA-256 hash.

split_scientific_name	<i>Split a scientific name into genus and epithets</i>
-----------------------	--

Description

Several sources publish a name without the parsed parts the unified schema carries, so the genus, specific epithet and infraspecific epithet are taken from the name itself.

Usage

split_scientific_name(name, genus = NULL)

Arguments

name Character vector of scientific names.

genus Character vector of known genus names (same length as name), or NULL to take the genus from the name.

Details

When genus is supplied and the name begins with it, the split is anchored there; otherwise the first word of the name is taken as the genus. Anchoring on a known genus rather than on a rank vocabulary means a rank spelling this function has never seen still resolves, and it keeps a two-word genus from being read as a binomial.

The infraspecific epithet is the last word following the specific epithet, so a name carrying a rank marker (*Poa annua* subsp. *exilis*) and a bare trinomial (*Larus fuscus graellsii*) both resolve to the epithet itself. A free-standing hybrid multiplication sign is dropped rather than read as an epithet, and so is a subgenus in parentheses after the genus.

The caller decides which rows to apply this to: a name above genus rank still returns its first word as genus, so a source that stores family and order rows in the same table gates on rank itself.

Value

A list of three character vectors: genus, specific and infraspecific, each the length of name. Absent parts are NA.

Examples

```
split_scientific_name("Poa annua subsp. exilis")
split_scientific_name("Larus fuscus graellsii")
split_scientific_name("Quercus", genus = "Quercus")
```

tsita_crosswalk

The enrichment-to-T-SITA crosswalk

Description

Loads inst/extdata/tsita_crosswalk.csv and resolves each tsita_label to its concept URI against `tsita_vocab()`, failing loudly if a label is not a known T-SITA concept. Each row maps one enrichment column onto a T-SITA concept: a row with an empty raw_value maps the column (the trait axis); a row with a raw_value maps one categorical value of that column onto an attribute concept.

Usage

```
tsita_crosswalk(enrichment = NULL)
```

Arguments

enrichment	Character or NULL. Restrict to one or more enrichment names; NULL (default) returns the whole crosswalk.
------------	--

Value

A data.frame with columns enrichment, column, raw_value (NA for an axis mapping), tsita_label, tsita_uri.

See Also

[tsita_vocab\(\)](#)

tsita_vocab	<i>The T-SITA thesaurus as a flat concept table</i>
-------------	---

Description

Loads the frozen T-SITA vocabulary (inst/extdata/tsita_vocab.csv): one row per concept, with its stable ARK URI, prefLabel, the top concept it sits under (Trait or Ecological_preference), its depth below that top, the broader parent URI, and the definition / scopeNote. prefLabel is unique across the thesaurus, which is what lets the crosswalk key on labels and resolve URIs from here. The result is cached for the session.

Usage

`tsita_vocab()`

Value

A data.frame with columns uri, id, prefLabel, top, depth, broader, definition, scopeNote.

See Also

[tsita_crosswalk\(\)](#)

update_enrichment_manifest	<i>Update manifest.json enrichment entry from built meta.json</i>
----------------------------	---

Description

Enrichment .vtr files are published under a single rolling release tag (e.g. enrichment-2026.06) regardless of the dataset's own version. The manifest entry therefore records two distinct versions: latest holds the rolling release version (matching the enrichment-<latest> download tag) while source_version holds the dataset's own version (from the meta sidecar). Pass release_version to set the release tag explicitly; when omitted it falls back to the dataset version (correct only when the two happen to coincide).

Usage

```
update_enrichment_manifest(  
  manifest_path,  
  name,  
  vtr_path,  
  release_version = NULL,  
  repo = "gcol33/taxifydb",  
  runtime = FALSE  
)
```

Arguments

manifest_path	Character. Path to manifest.json.
name	Character. Enrichment identifier.
vtr_path	Character. Path to the built .vtr file.
release_version	Character or NULL. Rolling release version used in the download URL tag (enrichment-<release_version>). Defaults to the dataset version from the meta sidecar.
repo	Character. GitHub repo for URL construction.
runtime	Logical. When TRUE, also populate the runtime-only fields the taxify door needs (trait_cols, species_col, static, source_format, citation) from the meta sidecar, filling only fields the entry does not already carry so hand-curated values are preserved. Two fields are exceptions, both because the stored value describes the build or the source rather than merely accompanying it. citation is rewritten when the build has moved to a different source_url or source_doi, since the stored text names the source it was written for; a stored citation that already names the entry's source_doi is kept, as it names the same work the new URL serves. trait_cols is always taken from the build (enrichment_trait_cols()), since the runtime exposes whatever the .vtr carries. Used when writing taxify's inst/manifest.json; left FALSE for this repo's lean build-side manifest. A new enrichment's runtime entry is then complete from the build alone, with no manual curation step.

Value

The updated manifest (invisibly).

update_manifest	<i>Update manifest.json after a successful backbone build</i>
-----------------	---

Description

latest is the release tag the download URL points at, and it is the source release the build recorded in its .meta sidecar (WFO 2026.06, OTT 3.7.3), so one field names both the download and the data.

Usage

```
update_manifest(
  manifest_path,
  backend_name,
  version,
  vtr_path,
  delta_path = NULL,
  delta_from = NULL,
  delta_from_content_id = NULL,
  extras = NULL,
  repo = "gcol33/taxifydb",
  source_url = NULL
)
```

Arguments

<code>manifest_path</code>	Character. Path to manifest.json.
<code>backend_name</code>	Character.
<code>version</code>	Character. The source release the build recorded in its <code>.meta</code> ; a different value is an error.
<code>vtr_path</code>	Character.
<code>delta_path</code>	Character or NULL.
<code>delta_from</code>	Character or NULL. Previous version the delta is from.
<code>delta_from_content_id</code>	Character or NULL. Content id (md5) of the exact build the delta was cut against, as <code>create_delta()</code> records it. The runtime applies the patch only to a local build with this id; a delta recorded without one is never applied, since a version label does not identify the bytes a patch needs.
<code>extras</code>	Character vector or NULL. Paths to sidecar artifacts uploaded with the release. Recorded as <code>extras: [{name, url, size, sha256}]</code> in the manifest entry; the runtime downloader fetches each into the same versioned directory as the main <code>.vtr</code> . NULL (the default) leaves an existing <code>extras</code> block untouched, since a sidecar is published on its own release cadence and its recorded URL names the tag it came from, not this one. Pass character (<code>0</code>) to remove the block.
<code>repo</code>	Character. GitHub repo for URL construction.
<code>source_url</code>	Character or NULL. Original data source URL. When NULL (the default), it is read from the <code>url</code> field of the <code>.meta</code> sidecar that <code>build_vtr()</code> writes next to <code>vtr_path</code> , which is the backend's single source of truth for provenance.

Value

The updated manifest (invisibly).

vtr_changed	<i>Does a build warrant a release?</i>
-------------	--

Description

Several backbones read a pinned or frozen source and rebuild to the same bytes every time – Euro+Med from a frozen snapshot release, GBIF from its last backbone, LCVP from a tagged data release. Publishing those again would re-upload a file every taxify user already holds. A build is therefore a change when its bytes differ from the published asset, or when the manifest records it under a different version than the source release the build names: the second case re-releases identical bytes under the tag that says what they are.

Usage

```
vtr_changed(manifest_path, backend_name, vtr_path, version = NULL)
```

Arguments

manifest_path	Character. Path to manifest.json.
backend_name	Character. Backend identifier.
vtr_path	Character. Path to the freshly built .vtr.
version	Character or NULL. The source release the build recorded; NULL compares bytes only.

Details

Fails open: anything that cannot be determined – no manifest, no entry for this backbone, no recorded hash, a first-ever build – counts as changed, so an uncertain case still publishes.

Value

Logical scalar. TRUE when a release is warranted.

wfo_latest_edition	<i>Resolve the newest WFO Plant List edition on Zenodo</i>
--------------------	--

Description

Zenodo answers a concept record with its latest version. The edition label that version carries (2026-06) is the release the backbone is built as.

Usage

```
wfo_latest_edition(verbose = TRUE)
```

Arguments

verbose Logical.

Value

A list with record (Zenodo record id), edition (as WFO labels it), version (YYYY.MM) and url (the Darwin Core backbone archive).

Index

.alien_first_record_sentinels, [81](#)
.anubis_fetch(), [58](#)
.dedup_keep_richest(), [162](#)
.gbif_col_names, [78](#)
.keep_earliest_first_record(), [81](#)
.read_leda_table(), [124](#)
.seebens_location_key(), [81](#)

apply_delta, [8](#)
assert_built_matches_registry, [8](#)
attach_references, [9](#)
attach_references(), [16](#), [85](#), [91](#), [105](#), [113](#),
[124](#)

build_algaebase, [9](#)
build_all_name_lookups, [10](#)
build_all_name_lookups(), [162](#)
build_avilist, [10](#)
build_backend, [11](#)
build_backend(), [12](#), [20](#), [25](#), [74](#)
build_backend_coverage, [11](#)
build_backend_coverage(), [25](#)
build_betsi_recovery, [12](#)
build_betsi_recovery(), [75](#), [89](#)
build_col, [13](#)
build_colxr, [13](#)
build_enrichment, [14](#)
build_enrichment(), [9](#), [12](#)
build_enrichment_vtr, [15](#)
build_enrichment_vtr(), [9](#), [14](#)
build_euromed, [17](#)
build_evc, [17](#)
build_fishbase, [18](#)
build_fungorum, [18](#)
build_gbif, [19](#)
build_gbif_occurrence_counts, [19](#)
build_genus_register, [20](#)
build_genus_register(), [25](#)
build_itis, [21](#)
build_lcvp, [21](#)

build_lpsn, [22](#)
build_mdd, [22](#)
build_meow, [23](#)
build_name_lookup, [23](#)
build_ncbi, [24](#)
build_ott, [24](#)
build_register, [25](#)
build_reptiledb, [26](#)
build_sealifebase, [26](#)
build_vegcz, [27](#)
build_vtr, [27](#)
build_vtr(), [28](#), [29](#)
build_vtr_streamed, [28](#)
build_vtr_streamed(), [51](#), [53](#)
build_wcvp, [30](#)
build_wfo, [30](#)
build_wgsrpd, [31](#)
build_wgsrpd(), [23](#)
build_worms, [31](#)

check_all_enrichment_versions, [32](#)
check_ckan_version, [32](#)
check_content_md5, [33](#)
check_crossref_version, [33](#)
check_datacite_version, [34](#)
check_dataone_version, [35](#)
check_dataverse_version, [35](#)
check_dryad_version, [36](#)
check_edc_version, [36](#)
check_enrichment_source_version, [37](#)
check_figshare_version, [38](#)
check_figshare_version(), [16](#), [145](#)
check_gbif_api_version, [38](#)
check_gift_version, [39](#)
check_github_version, [39](#)
check_ipt_version, [40](#)
check_kew_cvalues_version, [40](#)
check_last_modified, [41](#)
check_lcvp_version, [41](#)
check_ncbi_archive_version, [42](#)

- check_ott_version, [42](#)
- check_pangaea_version, [43](#)
- check_phaidra_version, [43](#)
- check_rfishbase_version, [44](#)
- check_seanoe_version, [44](#)
- check_wayback_version, [45](#)
- check_wst_version, [45](#)
- check_zenodo_version, [46](#)
- colxr_col_publications, [47](#)
- colxr_col_publications(), [78](#)
- colxr_gbif_crosswalk, [47](#)
- colxr_gbif_crosswalk(), [48](#), [78](#)
- colxr_gbif_lookup, [48](#)
- colxr_gbif_lookup(), [47](#)
- colxr_latest_release, [48](#)
- colxr_latest_release(), [56](#)
- colxr_request_export, [49](#)
- colxr_request_export(), [48](#)
- count_gbif_occurrences, [50](#)
- count_vtr_rows, [50](#)
- create_delta, [51](#)
- delim_chunk_feed, [51](#)
- delim_chunk_feed(), [53](#)
- delim_fk_lookup, [53](#)
- download_algaebase, [54](#)
- download_and_unzip, [54](#)
- download_avilist, [55](#)
- download_cf_file, [55](#)
- download_col, [56](#)
- download_colxr, [56](#)
- download_curl_file, [57](#)
- download_curl_file(), [58](#)
- download_dryad_file, [58](#)
- download_edi_file, [58](#)
- download_edi_file(), [36](#)
- download_euromed, [59](#)
- download_euromed(), [17](#)
- download_evc, [60](#)
- download_evc(), [18](#)
- download_fungorum, [60](#)
- download_gbif, [61](#)
- download_gbif_api_pages, [61](#)
- download_gbif_occurrence_counts, [62](#)
- download_groot, [62](#)
- download_itis, [63](#)
- download_lcvp, [63](#)
- download_lpsn, [64](#)
- download_mdd, [64](#)
- download_meow, [65](#)
- download_ncbi, [65](#)
- download_ott, [66](#)
- download_reptiledb, [66](#)
- download_reptiledb(), [156](#)
- download_vegcz, [67](#)
- download_vegcz(), [27](#)
- download_wcvp, [67](#)
- download_wfo, [68](#)
- download_wgsrpd, [68](#)
- download_wiley_supplement, [69](#)
- download_worms, [69](#)
- enrichment_emergency_fallback, [70](#)
- enrichment_trait_cols, [70](#)
- enrichment_trait_cols(), [168](#)
- gbif_count_keys, [71](#)
- gbif_count_keys(), [19](#)
- gbif_higher_lookup(), [78](#)
- gen_spe, [71](#)
- gen_spe(), [161](#)
- GIFT::GIFT_references(), [113](#)
- GIFT::GIFT_traits(), [113](#)
- gunzip_file, [72](#)
- harvest_ckan_datastore, [72](#)
- harvest_plazi_dwca, [73](#)
- harvest_plazi_dwca(), [134](#)
- has_xdelta3, [73](#)
- inrae_genus_dict, [74](#)
- list_backends, [74](#)
- list_backends(), [10](#), [11](#), [161](#), [163](#)
- list_betsi_recovery, [75](#)
- list_betsi_recovery(), [12](#), [89](#)
- list_build_only_enrichments, [75](#)
- list_enrichments, [76](#)
- list_enrichments(), [75](#)
- list_geometry, [76](#)
- list_geometry(), [11](#), [74](#)
- list_scrape_only_enrichments, [76](#)
- list_scrape_only_enrichments(), [75](#)
- normalize_backbone, [77](#)
- normalize_backbone(), [29](#), [145](#)
- normalize_colxr, [77](#)
- normalize_gbif, [78](#)
- normalize_wcvp, [79](#)

normalize_wfo, 79
normalize_worms, 80

parse_algae_traits, 80
parse_alien_first_records, 81
parse_alpine_carabids, 82
parse_amniote, 82
parse_amphibio, 83
parse_anage, 83
parse_animaltraits, 84
parse_arctic, 84
parse_arthropod_traits, 85
parse_austraits, 85
parse_avonet, 86
parse_bacdiv, 86
parse_baseflor, 87
parse_bee_ostwald, 87
parse_bet, 88
parse_betsi_collembola_body_length, 88
parse_betsi_recovery, 89
parse_betsi_recovery(), 75
parse_beukhof, 89
parse_bien, 90
parse_birdbase, 90
parse_blanchard, 91
parse_brot, 91
parse_bryoatt, 92
parse_ccdb, 92
parse_cefas_btrait, 93
parse_chelonians, 93
parse_chowdhury, 94
parse_clopla, 94
parse_combine, 95
parse_common_names, 95
parse_copepod_traits, 96
parse_coral_traits, 97
parse_diaz_traits, 97
parse_disperse, 98
parse_eberswalde, 98
parse_ecoflora, 99
parse_ecomorphosis, 100
parse_edwards_phyto, 100
parse_eive, 101
parse_ellers_collembola, 101
parse_elton_traits, 102
parse_epa_freshwater, 103
parse_eupolltrait, 103
parse_eurobat, 104
parse_euromed_distribution, 104

parse_faprotax, 105
parse_finand, 106
parse_fishbase, 106
parse_fishbase(), 137
parse_fishmorph, 107
parse_fishtraits, 107
parse_floraweb, 108
parse_frugivoria, 109
parse_fungal_traits, 109
parse_fungalroot, 110
parse_funguild, 110
parse_fw_insects_conus, 111
parse_gbif_common_names, 111
parse_gidias, 112
parse_gift, 113
parse_globi, 114
parse_globtherm, 114
parse_glonaf, 115
parse_gmpd, 115
parse_griis, 116
parse_groot, 116
parse_groot(), 62
parse_gwdd, 117
parse_homerange, 117
parse_hosts, 118
parse_huang_amph, 118
parse_hydraulics, 119
parse_imageomics_neon, 120
parse_invacost, 120
parse_italic, 121
parse_iucn, 122
parse_kew_cvalues, 122
parse_kew_sid, 123
parse_leda, 124
parse_leptrait, 125
parse_madin, 125
parse_marine_distribution, 126
parse_monograph_body_length, 126
parse_monograph_body_length(), 127
parse_monograph_collembola, 127
parse_ncbi_common_names, 128
parse_nesttrait, 128
parse_noddb, 129
parse_nztd, 129
parse_octocoral, 130
parse_odonata, 130
parse_ott_common_names, 131
parse_pantheria, 131

- parse_parravicini, 132
- parse_pelagic, 132
- parse_phylacine, 133
- parse_plantatt, 133
- parse_plazi_collembola_body_length, 134
- parse_pottier, 134
- parse_quimbayo, 135
- parse_ramond, 135
- parse_repttraits, 136
- parse_rimet_phyto, 136
- parse_saproxylic, 137
- parse_sealifebase, 137
- parse_sharkipedia, 138
- parse_sheld, 138
- parse_spider_traits, 139
- parse_sworm, 139
- parse_tetradensity, 140
- parse_thermofresh, 140
- parse_tree_of_sex, 141
- parse_usda_fungus_host, 142
- parse_useful_plants, 142
- parse_virion, 143
- parse_wcvp, 143
- parse_zanne, 144
- parse_zooplankton, 144
- precompute_backbone, 145
- precompute_backbone(), 29, 78, 79, 148, 149, 152–155, 157, 158
- probe_upstream_identity, 145
- probe_upstream_identity(), 16, 37
- publish_enrichment_release, 146
- publish_release, 147
- read_algaebase, 148
- read_avilist, 148
- read_col, 149
- read_colxr, 149
- read_colxr(), 77
- read_dmp, 150
- read_euromed, 150
- read_evc, 151
- read_evc(), 157
- read_fungorum, 151
- read_gbif, 152
- read_gbif(), 78
- read_gbif_occurrence_counts, 152
- read_gbif_occurrence_counts(), 78, 152
- read_itis, 153
- read_lcvp, 153
- read_lpsn, 154
- read_mdd, 154
- read_meow, 155
- read_ncbi, 155
- read_ott, 156
- read_reptiledb, 156
- read_vegcz, 157
- read_wcvp, 157
- read_wcvp(), 79
- read_wfo, 158
- read_wfo(), 79
- read_wgsrpd, 158
- read_wgsrpd(), 155
- read_worms, 159
- read_worms(), 80
- register_backbones, 159
- register_backbones(), 11, 12, 20, 25
- resolve_betsi_codes, 160
- resolve_betsi_codes(), 74
- resolve_enrichment_names, 161
- resolve_enrichment_names(), 14, 15, 163
- resolve_hierarchy, 162
- resolve_name_map, 163
- resolve_name_map(), 161, 162
- rfishbase_release, 164
- sha256, 165
- split_scientific_name, 165
- taxify::backbone_fixed_kingdom(), 163
- taxify::normalize_kingdom_group(), 163
- taxify::taxify(), 162
- tsita_crosswalk, 166
- tsita_crosswalk(), 167
- tsita_vocab, 167
- tsita_vocab(), 166, 167
- update_enrichment_manifest, 167
- update_manifest, 168
- utils::read.delim(), 52
- vtr_changed, 170
- wfo_latest_edition, 170
- wfo_latest_edition(), 30, 68