Package: vegbankr (via r-universe)

May 26, 2026

Title Interface to the 'VegBank' Vegetation Plot Database

Version 1.0.0

Date 2026-03-26

Description Provides a programmatic interface to 'VegBank', the vegetation plot database of the Ecological Society of America's Panel on Vegetation Classification, hosted by the National Center for Ecological Analysis and Synthesis (<https://www.nceas.ucsb.edu>). 'VegBank' contains vegetation plot data, community types recognized by the U.S. National Vegetation Classification and others, and all 'ITIS/USDA' plant taxa along with other taxa recorded in plot records. As a 'VegBank' API client, the 'vegbankr' package currently supports querying and downloading vegetation plot records and other supporting information from the 'VegBank' database, and validating and uploading new data to the 'VegBank' database as well.

License Apache License (>= 2) | file LICENSE

URL https://github.com/NCEAS/vegbankr, https://nceas.github.io/vegbankr/, https://nceas.r-universe.dev/vegbankr

BugReports https://github.com/NCEAS/vegbankr/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports DBI, base64enc, cli, curl, dplyr, duckdb, httr2, jsonlite, nanoparquet, purrr, rlang, tidyr

Suggests DT, knitr, httptest2, pkgdown, rmarkdown, spelling, testthat (>= 3.0.0), withr

Depends R (>= 4.1.0)

Language en-US

Config/testthat/edition 3

Config/spelling/custom_dicts inst/WORDLIST

X-CRAN-Comment Spelling: ITIS NCEAS VegBank vegbankr

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev libssl-dev xz-utils

Repository https://nceas.r-universe.dev

Date/Publication 2026-03-27 01:29:58 UTC

RemoteUrl https://github.com/NCEAS/vegbankr

RemoteRef HEAD

RemoteSha 67b455247383d0d0c871ea2b9a0ece8001d6770b

Contents

canonicalize_names *Canonicalize VegBank column names (i.e. convert to snake_case), using a package-provided lookup table by default*

Description

Takes a data frame of VegBank records, and canonicalizes the column names by converting them to snake_case via a package-provided lookup_table. If any column names in the input data frame are unmatched in the lookup table, these are left unaltered in the output, and a warning message is displayed.

Usage

```
canonicalize_names(target_df, lookup_df)
```

Arguments

target_df	(dataframe)
lookup_df	(dataframe) Optional custom names lookup table

Details

Callers may optionally provide a lookup table

Value

A data frame matching the input data frame, but with canonicalized column names

Examples

```
canonicalize_names(data.frame(
  stratum_ID = integer(),
  stratummethodname = character()
))
```

get_page_details *Get paging details for a VegBank dataframe*

Description

Reports paging details associated with a dataframe produced by querying an VegBank API endpoint with limit/offset-based pagination.

Usage

```
get_page_details(x)
```

Arguments

x	Dataframe, presumably returned by a VegBank getter

Value

Named vector with the following elements, if available (any of these values not attached to the data frame will simply be missing from the returned vector):

- count_reported: the full record count reported by the API
- offset: the record offset used in the API query
- limit: the record limit used in the API query
- count_returned: the actual count of returned records

Examples

```
## Not run:
parties <- get_all_parties(limit=10, offset=50)
get_page_details(parties)

## End(Not run)
```

---

validate_at_least_one_present

Validate at least one of two columns is present

Description

Validate at least one of two columns is present

Usage

```
validate_at_least_one_present(df, col1, col2)
```

Arguments

df	Data frame to validate
col1	First column name
col2	Second column name

Value

Logical. TRUE if validation passes, FALSE otherwise

`validate_no_duplicates` *Validate no duplicate values in specified columns*

Description

Validate no duplicate values in specified columns

Usage

```
validate_no_duplicates(df, columns)
```

Arguments

`df`	Data frame to validate
`columns`	Character vector of column names to check

Value

Logical. TRUE if validation passes, FALSE otherwise

`validate_no_nulls` *Validate no NULL values in specified columns*

Description

Validate no NULL values in specified columns

Usage

```
validate_no_nulls(df, columns)
```

Arguments

`df`	Data frame to validate
`columns`	Character vector of column names to check

Value

Logical. TRUE if validation passes, FALSE otherwise

`validate_values_exist` *Validate values exist in parent table*

Description

Validate values exist in parent table

Usage

```
validate_values_exist(
  child_df,
  child_col,
  parent_df,
  parent_col,
  optional = FALSE
)
```

Arguments

`child_df`	Child data frame
`child_col`	Column name in child data frame
`parent_df`	Parent data frame
`parent_col`	Column name in parent data frame
`optional`	(Boolean) If FALSE, check will fail if parent column is not found. Otherwise, check will skip.

Value

Logical. TRUE if validation passes, FALSE otherwise

`vb_access_token_is_valid` *Check whether the stored access token is present and not expired*

Description

Decodes the `exp` claim from the stored JWT access token and returns `TRUE` if the token exists and has a valid expiry.

Usage

```
vb_access_token_is_valid()
```

Value

Logical TRUE if the access token is present and valid, FALSE otherwise.

See Also

vb_set_token(), vb_refresh_tokens()

Examples

```
## Not run:
vb_access_token_is_valid()

## End(Not run)
```

---

vb_count *Get record count for a VegBank resource*

Description

vb_count() and friends (implemented for each of the VegBank resource types) are used to retrieve a record count from the VegBank API, optionally constrained by filtering query parameters. The count is the same as the count_reported attribute attached to API data responses (see get_page_details()), but is returned here without actually querying for any data.

Usage

```
vb_count(resource, ...)

vb_count_community_classifications(...)

vb_count_community_concepts(...)

vb_count_cover_methods(...)

vb_count_named_places(...)

vb_count_parties(...)

vb_count_plant_concepts(...)

vb_count_plot_observations(...)

vb_count_projects(...)

vb_count_references(...)

vb_count_roles(...)
```

```
vb_count_stem_counts(...)

vb_count_taxon_importances(...)

vb_count_strata(...)

vb_count_stratum_methods(...)

vb_count_taxon_observations(...)

vb_count_user_datasets(...)
```

Arguments

resource	VegBank API resource (e.g., plot-observation)
...	Additional API query parameters

Value

Integer count, or NULL if the count cannot be retrieved

Examples

```
## Not run:
vb_count("projects")
vb_count_projects(search="california")

## End(Not run)
```

---

vb_create_dataset *Create a VegBank dataset*

Description

Use the VegBank API to create a user dataset, which defines a collection of plot observations that can be cited in VegBank.

Usage

```
vb_create_dataset(name, description, observations, dry_run = FALSE)
```

Arguments

name	A single character string giving the dataset name. Must be 100 characters or fewer.
description	A single character string describing the dataset.
observations	A character vector of observation codes. Each element must match the pattern "ob.<positive_integer>" (e.g. "ob.2948"), and must refer to a plot observation in VegBank. The vector must not be empty.
dry_run	Logical indicating whether to perform a dry run. If TRUE, the API will validate the data without committing changes to the database. Default is FALSE.

Value

The processed response object from the VegBank API documenting what (if anything) was successfully created in VegBank.

Examples

```
## Not run:
vb_create_dataset(
  name = "Test Dataset 001",
  description = "A test dataset containing 10 observations",
  observations = paste0("ob.", 2948:2957)
)

## End(Not run)
```

---

vb_debug *Enable VegBank API debugging mode*

Description

Set VegBank debug level used to send API requests. This currently controls two things:

1. Verbosity of API requests, specifically as handled by `httr::req_perform()`
2. Reporting of the elapsed time, as a console message

Usage

```
vb_debug(verbosity = 1)
```

Arguments

verbosity	Integer between 0-3 (Default: 1). Using 0 is equivalent to setting `vb_undebug()`, in which case no information is reported. Values between 1-3 control verbosity level passed to `httr2::req_perform()`, and in all cases include display of API request time duration.

See Also

vb_undebug(), httr2::req_perform()

---

vb_get *Retrieve data from VegBank*

Description

Retrieve data from the VegBank REST API. vb_get() is the core function that can access any resource type. Resource-specific functions like vb_get_plot_observations() and vb_get_plant_concepts() are convenience wrappers that provide resource-appropriate defaults. For typical usage, the resource-specific functions should be preferred, but vb_get() may be useful when using vegbankr in a R package or leveraging any experimental VegBank read API features that may not (yet) supported by vegbankr.

Usage

```
vb_get_projects(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  sort = NULL,
  ...
)

vb_get_parties(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  sort = NULL,
  ...
)

vb_get_roles(vb_code = NULL, limit = 100, offset = 0, parquet = NULL, ...)

vb_get_references(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  sort = NULL,
```

```
  ...
)

vb_get_named_places(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  ...
)

vb_get_cover_methods(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  with_nested = NULL,
  ...
)

vb_get_stratum_methods(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  with_nested = NULL,
  ...
)

vb_get_plant_concepts(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  status = NULL,
  with_nested = NULL,
  ...
)

vb_get_taxon_observations(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
```

```
  detail = NULL,
  with_nested = NULL,
  ...
)

vb_get_taxon_importances(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  detail = NULL,
  with_nested = NULL,
  ...
)

vb_get_stem_counts(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  detail = NULL,
  ...
)

vb_get_strata(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  detail = NULL,
  ...
)

vb_get_taxon_interpretations(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  detail = NULL,
  ...
)

vb_get_community_concepts(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
```

```
  status = NULL,
  with_nested = NULL,
  ...
)

vb_get_community_classifications(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  detail = NULL,
  with_nested = NULL,
  ...
)

vb_get_community_interpretations(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  detail = NULL,
  with_nested = NULL,
  ...
)

vb_get_plot_observations(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  search = NULL,
  status = NULL,
  detail = NULL,
  with_nested = NULL,
  num_taxa = NULL,
  num_comms = NULL,
  ...
)

vb_get_user_datasets(
  vb_code = NULL,
  limit = 100,
  offset = 0,
  parquet = NULL,
  ...
)
```

```
vb_get(
  resource,
  vb_code = NULL,
  by = NULL,
  parquet = TRUE,
  clean_names = FALSE,
  limit = 100,
  offset = 0,
  ...
)
```

Arguments

vb_code	Optional VegBank code to retrieve a specific record.
limit	Integer specifying maximum number of records to return. Default is 100. Set to NULL to use API default.
offset	Integer specifying the offset for pagination. Default is 0. Set to NULL to use API default.
parquet	Logical indicating whether to request data in Parquet format instead of JSON. This is transparent to users insofar as data will be returned as a data frame in either case, but performance may differ between the two options. If not specified, defaults to TRUE for collection queries and FALSE for single-record queries.
search	Optional search string for filtering results based on full-text search or the resource's vb_code (the latter of which is functionally equivalent to using the vb_code argument directly). Available for: plot-observations, plant-concepts, community-concepts, projects, and parties.
sort	Optional string for sorting results. Prepend with "-" for descending order (e.g., "-obs_count"). Available for: • **plot-observations:** "default", "author_obs_name" • **plant-concepts:** "default", "plant_name", "obs_count" • **community-concepts:** "default", "comm_name", "obs_count" • **projects:** "default", "project_name", "obs_count" • **parties:** "default", "surname", "organization_name", "obs_count"
...	Additional query parameters passed to the API endpoint as key-value pairs. E.g., foo="bar" will add a URL query parameter "?foo=bar" to the API GET request.
with_nested	Logical indicating whether to include nested data structures. All endpoints support FALSE. For those that support TRUE, this is the default for individual record queries, otherwise the default is FALSE. Set to NULL to use the API default.
status	Optional string for limiting results by status, with default equivalent to 'any' if unset. Available for: • **plot-observations:** "any", "current" • **community-concepts:** "any", "current", "accepted", "current_accepted"
detail	Character string specifying level of detail. All endpoints support "full" detail. For those that support "minimal" detail, this is the default for collection queries, otherwise the default is "full". Plot observations additionally support detail="geo". In all cases, set to NULL to use the API default.

num_taxa — *Available only for plot-observations.* Integer specifying maximum number of taxon observations to return as a nested array. If not specified, defaults to 5 for collection queries and 5000 for single-record queries. Set to NULL to use the API default.

num_comms — *Available only for plot-observations.* Integer specifying maximum number of community classifications records to return as a nested array. If not specified, defaults to 5 for collection queries and 5000 for single-record queries. Set to NULL to use the API default.

resource — *Available only for* vb_get(). Character string specifying the VegBank resource type to retrieve (e.g., "plot-observations", "projects").

by — *Available only for* vb_get(). Character string specifying the VegBank resource type associated with the provided vb_code (if specified). If omitted, this will be inferred based on the format of the vb_code, so in general it is not necessary. However, if using vb_get() in a function, specifying by may be useful to ensure that the correct resource is being queried.

clean_names — *Available only for* vb_get(). Logical indicating whether to canonicalize column names to snake case. Default is FALSE.

Details

The vb_get*() family of functions can all be used to perform three types of queries, which differ in terms of the set of records returned as well as the default query parameters applied.

Available resource-specific functions::

- vb_get_plot_observations() - Plot observation data
- vb_get_community_concepts() - Community concepts (assertions) linked to community names through usages
- vb_get_community_classifications() - Community classification events wherein one or more community concepts were applied to a plot observation
- vb_get_community_interpretations() - Assignments of community names and authorities (i.e., community concepts) to specific plot observations, as part of a community classification event
- vb_get_plant_concepts() - Plant concepts (assertions) linked to plant names through usages
- vb_get_taxon_observations() - Data provider's determination of taxa observed on a plot, and the overall cover of those taxa
- vb_get_taxon_importances() - Cover/etc, and optionally stem counts, of taxa observed on a plot, overall and by stratum if so defined
- vb_get_stem_counts() - Stem counts of observed taxa, overall and by stratum if so defined
- vb_get_strata() - Defined strata in which taxon importance and/or stem counts have been recorded within a given plot observation, consistent with some stratum method
- vb_get_taxon_interpretations() - Assignments of taxon names and authorities (i.e., plant concepts) to specific taxon observations
- vb_get_cover_methods() - Information about registered coverclass methods
- vb_get_stratum_methods() - Information about registered strata sampling protocols
- vb_get_named_places() - Information about named places registered in VegBank

- vb_get_references() - Information about references cited within VegBank
- vb_get_projects() - Information about projects established to collect vegetation plot data
- vb_get_parties() - Information about people and organizations who have contributed to the collection or interpretation of a plot
- vb_get_roles() - VegBank-defined role types, which can be used to define the role of a party contributing to a plot observation, classification, etc
- vb_get_user_datasets() - Information about people and organizations who have contributed to the collection or interpretation of a plot

Query types:

Single-record queries:
A query is considered a "single-record query" if a vb_code is provided with prefix that matches the type of the target resource. For example, vb_get_plot_observations("ob.2948") will return the single VegBank plot observation record corresponding to "ob.2948", if one exists.

Cross-resource collection queries:
A query is considered a "cross-resource collection query" if a vb_code is provided with prefix matching a resource type that *differs* from the target resource. For example, vb_get_plot_observations("pj.340"), will return the collection of VegBank plot observation records corresponding to project pj.340, if the project and corresponding plot observations exist.

Full collection queries:
A query is considered a "full collection query" if no vb_code is provided. For example, vb_get_plot_observations() will return all plot observations. Both collection query types are paginated via limit and offset, and can be further filtered with parameters like search when supported.

Smart defaults::

Many parameters have "smart defaults" that change based on whether you're querying a single record or a collection of records. In general, single-record queries automatically use settings optimized for detailed individual records (more nested data, higher limits for related entities), whereas collection queries use leaner settings. For example, vb_get_plot_observations() uses detail = "minimal" and with_nested = FALSE for collection queries, but detail = "full" and with_nested = TRUE for single records.

To use the API's own defaults instead of these smart defaults, explicitly pass NULL for the parameter (e.g., detail = NULL).

Value

A data frame containing the requested VegBank data.

Examples

```
## Not run:
# Collection query - uses minimal detail and excludes nested fields
plots <- vb_get_plot_observations(limit = 10)

# Single record - uses full detail with nested fields
plot <- vb_get_plot_observations(vb_code = "ob.12345")

# Override defaults explicitly
```

```
plots <- vb_get_plot_observations(limit = 10, detail = "full", num_taxa = 50)

# Use API defaults instead of smart defaults
plots <- vb_get_plot_observations(limit = NULL, detail = NULL, num_taxa = NULL)

# Get projects (different available parameters)
projects <- vb_get_projects(search = "heritage", limit = 20)

## End(Not run)
```

`vb_get_base_url` *Get the currently configured base URL for the VegBank API*

Description

Gets the base URL for the VegBank API. If previously set by the user, e.g. via `vb_set_base_url()`, this will be pulled from the global option (`vegbank.base_api_url`). If this option is unset, the package default value of `https://api.vegbank.org` will be used.

Usage

```
vb_get_base_url()
```

Value

A length-one character vector containing the base URL string

See Also

`vb_set_base_url()`

`vb_get_by_id` *Retrieve a VegBank resource by identifier*

Description

Fetches a VegBank resource using any supported identifier. This function first resolves the identifier to determine the resource type and internal code, then retrieves the full resource data.

Usage

```
vb_get_by_id(identifier, ..., verbose = FALSE)
```

Arguments

identifier	A character string specifying the VegBank identifier. This can be an accession code, DOI, or other supported identifier type.
...	Additional query parameters passed to vb_get() as key-value pairs. E.g., foo="bar" will add a URL query parameter "?foo=bar" to the API GET request.
verbose	Logical. If TRUE, prints a message indicating which resource was retrieved. Default is FALSE.

Value

A data frame containing the requested resource data. The structure depends on the resource type.

See Also

vb_resolve() for identifier resolution details

Examples

```
## Not run:
# Retrieve a dataset silently
data <- vb_get_by_id("VB.Ob.2948.ACAD143")

# Retrieve with informational message
data <- vb_get_by_id("VB.Ob.2948.ACAD143", verbose = TRUE)

## End(Not run)
```

---

vb_overview *Retrieve VegBank summary stats*

Description

Gets basic summary stats from VegBank as a list of data frames, including a table of high level counts and a few tables with "top N" (by count) or "latest N" (by upload date) reports.

Usage

```
vb_overview(limit = 5)
```

Arguments

limit	Integer specifying maximum number of records to return in the "top_n" and "latest_n" tables. Default is 5. Set to NULL to use API default.

Value

A list of data frames corresponding to the returned summary tables

Examples

```
## Not run:
# Retrieve summary stats
stats_list <- vb_overview(limit=5)

## End(Not run)
```

`vb_refresh_token_is_valid` *Check whether the stored refresh token is present and not expired*

Description

Decodes the `exp` claim from the stored JWT refresh token and returns `TRUE` if the token exists and has a valid expiry.

Usage

```
vb_refresh_token_is_valid()
```

Value

Logical `TRUE` if the refresh token is present and valid, `FALSE` otherwise.

See Also

`vb_set_token()`, `vb_refresh_tokens()`

Examples

```
## Not run:
vb_refresh_token_is_valid()

## End(Not run)
```

`vb_refresh_tokens` *Refresh stored Bearer tokens*

Description

Calls the VegBank `/refresh` endpoint with the currently stored refresh token to obtain a new access token and refresh token, then stores both via `vb_set_token()`.

Usage

```
vb_refresh_tokens()
```

See Also

vb_set_token(), vb_unset_token()

Examples

```
## Not run:
vb_refresh_tokens()

## End(Not run)
```

---

vb_resolve *Resolve a VegBank identifier*

Description

Queries the VegBank API to resolve a public identifier (such as an accession code or DOI) to its internal resource details.

Usage

```
vb_resolve(identifier)
```

Arguments

identifier	A character string specifying the VegBank identifier to resolve. This can be an accession code (e.g., "VB.Ob.2948.ACAD143") or other supported identifier type.

Value

A list containing the resolved identifier details with the following components:

identifier_value The original identifier value provided

identifier_type Type of identifier (e.g., "accession_code")

vb_code VegBank code for the resource

vb_resource_type VegBank resource type

Examples

```
## Not run:
# Resolve an accession code
result <- vb_resolve("VB.Ob.2948.ACAD143")
result$vb_code  # "ob.2948"

## End(Not run)
```

vb_set_base_url *Set a base URL for the VegBank API*

Description

Sets a global option specifying the base URL for the VegBank API using the provided `vb_base_url` string, optionally adding a port as "http(s)://vb_base_url:port".

Usage

```
vb_set_base_url(vb_base_url, port)
```

Arguments

`vb_base_url`	(character) The base URL, including protocol and domain
`port`	(numeric) Optional port value

See Also

`vb_get_base_url()`

Examples

```
vb_set_base_url("https://api.vegbank.org")
vb_set_base_url("http://localhost", port = 8080)
```

vb_set_token *Set Bearer token(s) for authenticated API requests*

Description

Stores an OAuth2 access token and/or refresh token in global options for use in subsequent authenticated API requests. Use `vb_unset_token()` to clear previously stored tokens.

Usage

```
vb_set_token(access_token = NULL, refresh_token = NULL, tokens = NULL)
```

Arguments

`access_token`	(character) Access token string.
`refresh_token`	(character) Refresh token string.
`tokens`	(list | character) Named list or JSON string containing `access_token` and/or `refresh_token`. Cannot be combined with `access_token` or `refresh_token`.

Details

Two input modes are supported — use one or the other, not both:

- **Individual strings:** pass `access_token`, `refresh_token`, or both.
- **Token dict:** pass `tokens` as a named list or JSON string with `access_token` and/or `refresh_token` keys.

See Also

`vb_unset_token()`, `vb_refresh_tokens()`

Examples

```
vb_set_token(access_token = "eyJhbGciOiJIUzI1NiJ9...")
vb_set_token(access_token = "eyJ...", refresh_token = "eyJ...")
vb_set_token(tokens = list(access_token = "eyJ...", refresh_token = "eyJ..."))
vb_set_token(tokens = '{"access_token": "eyJ...", "refresh_token": "eyJ..."}')
```

---

vb_undebug *Disable VegBank API debugging mode*

Description

Unset VegBank debugging. Equivalent to `vb_debug(0)`.

Usage

```
vb_undebug()
```

See Also

`vb_debug()`

---

vb_unset_token *Clear stored Bearer tokens*

Description

Removes the OAuth2 access token and refresh token previously stored by `vb_set_token()`. After calling this function, subsequent API requests will be sent without an `Authorization: Bearer` header.

Usage

```
vb_unset_token()
```

See Also

vb_set_token()

Examples

```
vb_unset_token()
```

---

vb_upload *Upload data to VegBank*

Description

Upload data to VegBank via the REST API. vb_upload() is the core function that can upload to any of the supported POST endpoints. Resource-specific functions like vb_upload_plot_observations() are convenience wrappers that apply resource-specific validation. For typical usage, the resource-specific functions should be preferred.

Usage

```
vb_upload(resource, ..., query_params = NULL, dry_run = FALSE)

vb_upload_plot_observations(
  plot_observations,
  projects = NULL,
  parties = NULL,
  references = NULL,
  soils = NULL,
  disturbances = NULL,
  community_classifications = NULL,
  strata = NULL,
  strata_cover_data = NULL,
  stem_data = NULL,
  taxon_interpretations = NULL,
  contributors = NULL,
  dry_run = FALSE
)

vb_upload_plant_concepts(
  plant_concepts,
  plant_names = NULL,
  plant_correlations = NULL,
  parties = NULL,
  references = NULL,
  what_to_deactivate = NULL,
  dry_run = FALSE
)
```

```
vb_upload_community_concepts(
  community_concepts,
  community_names = NULL,
  community_correlations = NULL,
  parties = NULL,
  references = NULL,
  what_to_deactivate = NULL,
  dry_run = FALSE
)

vb_upload_taxon_interpretations(
  taxon_interpretations,
  parties = NULL,
  references = NULL,
  dry_run = FALSE
)

vb_upload_community_classifications(
  community_classifications,
  parties = NULL,
  references = NULL,
  contributors = NULL,
  dry_run = FALSE
)

vb_upload_cover_methods(cover_methods, references = NULL, dry_run = FALSE)

vb_upload_stratum_methods(stratum_methods, references = NULL, dry_run = FALSE)
```

Arguments

resource	*Available only for* vb_upload(). Character string specifying the VegBank resource type to upload (e.g., "plot-observations", "projects").
...	Named data frames to upload. Each data frame should correspond to a table expected by the VegBank API for the specified resource. All arguments must be named, and at least one data frame must be provided.
query_params	A named list of parameter names and values to pass to the API as query parameters. For example, list(foo="bar") will be treated as ?foo=bar in the request URL.
dry_run	Logical indicating whether to perform a dry run. If TRUE, the API will validate the data without committing changes to the database. Default is FALSE.
plot_observations	A data frame containing details about plot observations
projects	A data frame containing details about new projects
parties	A data frame containing details about new parties
references	A data frame containing details about new references
soils	A data frame containing details about observed soils

`disturbances` A data frame containing details about observed disturbances

`community_classifications`
A data frame containing community classifications of observed plots

`strata` A data frame containing details about strata defined for plot observations

`strata_cover_data`
A data frame containing details about plant cover as observed in different strata of a plot

`stem_data` A data frame containing details about counts and/or other details about stems observed on a plot

`taxon_interpretations`
A data frame containing taxon interpretations of plants observed on a plot

`contributors` A data frame associating parties with their contributions to plot observations, projects, and/or community classifications

`plant_concepts` A data frame containing plant concepts as plant names associated with references, along with with status details and taxonomic parents

`plant_names` A data frame containing plant name usages associated with specific classification systems for new plant concepts

`plant_correlations`
A data frame defining correlations between plant concepts

`what_to_deactivate`
Available only for `vb_upload_plant_concepts()` *and* `vb_upload_community_concepts()`. Character string specifying what existing concepts to deactivate in VegBank. Supported values are "none" and "by_party", along with "by_party_below_order" for plant uploads only. VegBank default is `none`.

`community_concepts`
A data frame containing community concepts as community names associated with references, along with with status details and taxonomic parents

`community_names`
A data frame containing community name usages associated with specific classification systems for new community concepts

`community_correlations`
A data frame defining correlations between community concepts

`cover_methods` A data frame containing cover methods and their associated component cover indexes

`stratum_methods`
A data frame containing stratum methods and their associated component stratum types

Details

The `vb_upload*()` family of functions can all be used to upload data to VegBank queries, with each one differing in terms of what input dataframes are expected (and, in some case, required).

Available resource-specific functions::

1. `vb_upload_plot_observations()` - Plot observational data, including soil and disturbance observations, plant taxon observations and importance assessments (potentially including stem-level details) within any defined strata, and both individual plant taxon and overall vegetation community interpretation.
2. `vb_upload_plant_concepts()` - Plant concepts linked to plant names through usages, with some status designation
3. `vb_upload_community_concepts()` - Community concepts linked to community names through usages, with some status designation
4. `vb_upload_taxon_interpretations()` - Re-interpretation of existing observation), associating them with one or more VegBank plant concepts
5. `vb_upload_community_classifications()` - Re-interpretation of existing VegBank plot observations, associating them with one or more VegBank community concepts as part of a classification activity
6. `vb_upload_cover_methods()` - New cover methods, including all component cover indexes defined by the method
7. `vb_upload_stratum_methods()` - New stratum methods, including all component stratum types defined by the method

If `vb_debug()` is enabled, additional debugging details will be reported to the console, primarily focused on the data being uploaded.

Value

The processed response object from the VegBank API documenting what (if anything) was successfully uploaded to VegBank.

`vb_validate_community_concepts` *Validate VegBank loader tables for new community concepts*

Description

Performs validation checks on VegBank loader tables to ensure data integrity before upload. Validates required fields, uniqueness constraints, and referential integrity between related tables. Prints validation errors and warnings. This validation tool is a first pass at catching errors - full validation is only done at upload.

Usage

```
vb_validate_community_concepts(
  community_concepts,
  community_names = NULL,
  community_correlations = NULL,
  parties = NULL,
  references = NULL
)
```

Arguments

community_concepts	A data frame containing community concepts as community names associated with references, along with with status details and taxonomic parents
community_names	A data frame containing community name usages associated with specific classification systems for new community concepts
community_correlations	A data frame defining correlations between community concepts
parties	A data frame containing details about new parties
references	A data frame containing details about new references

Value

A named list with one element per table, each containing a logical value (TRUE if all validations passed for that table, FALSE otherwise). For example: `list(community_concepts = TRUE, community_names = FALSE, community_correlations = TRUE)`

---

vb_validate_cover_methods

Validate VegBank loader tables for new cover methods

Description

Performs validation checks on VegBank loader tables to ensure data integrity before upload. Validates required fields, uniqueness constraints, and referential integrity between related tables. Prints validation errors and warnings. This validation tool is a first pass at catching errors - full validation is only done at upload.

Usage

```
vb_validate_cover_methods(cover_methods, references = NULL)
```

Arguments

cover_methods	A data frame containing cover methods and their associated component cover indexes
references	A data frame containing details about new references

Value

A named list with one element per table, each containing a logical value (TRUE if all validations passed for that table, FALSE otherwise). For example: `list(community_concepts = TRUE, community_names = FALSE, community_correlations = TRUE)`

vb_validate_plant_concepts *Validate VegBank loader tables for new plant concepts*

Description

Performs validation checks on VegBank loader tables to ensure data integrity before upload. Validates required fields, uniqueness constraints, and referential integrity between related tables. Prints validation errors and warnings. This validation tool is a first pass at catching errors - full validation is only done at upload.

Usage

```
vb_validate_plant_concepts(
  plant_concepts,
  plant_names = NULL,
  plant_correlations = NULL,
  parties = NULL,
  references = NULL
)
```

Arguments

plant_concepts	A data frame containing plant concepts as plant names associated with references, along with with status details and taxonomic parents
plant_names	A data frame containing plant name usages associated with specific classification systems for new plant concepts
plant_correlations	A data frame defining correlations between plant concepts
parties	A data frame containing details about new parties
references	A data frame containing details about new references

Value

A named list with one element per table, each containing a logical value (TRUE if all validations passed for that table, FALSE otherwise). For example: `list(parties = TRUE, contributors = FALSE, plot_observations = TRUE)`

vb_validate_plot_observations *Validate VegBank loader tables for plot observations and related data*

Description

Performs validation checks on VegBank loader tables to ensure data integrity before upload. Validates required fields, uniqueness constraints, and referential integrity between related tables. Prints validation errors and warnings. This validation tool is a first pass at catching errors - full validation is only done at upload.

Usage

```
vb_validate_plot_observations(
  plot_observations,
  projects = NULL,
  parties = NULL,
  references = NULL,
  soils = NULL,
  disturbances = NULL,
  community_classifications = NULL,
  strata = NULL,
  strata_cover_data = NULL,
  stem_data = NULL,
  taxon_interpretations = NULL,
  contributors = NULL
)
```

Arguments

plot_observations	A data frame containing details about plot observations
projects	A data frame containing details about new projects
parties	A data frame containing details about new parties
references	A data frame containing details about new references
soils	A data frame containing details about observed soils
disturbances	A data frame containing details about observed disturbances
community_classifications	A data frame containing community classifications of observed plots
strata	A data frame containing details about strata defined for plot observations
strata_cover_data	A data frame containing details about plant cover as observed in different strata of a plot
stem_data	A data frame containing details about counts and/or other details about stems observed on a plot

`taxon_interpretations`
: A data frame containing taxon interpretations of plants observed on a plot

`contributors`
: A data frame associating parties with their contributions to plot observations, projects, and/or community classifications

Value

A named list with one element per table, each containing a logical value (TRUE if all validations passed for that table, FALSE otherwise). For example: `list(parties = TRUE, contributors = FALSE, plot_observations = TRUE)`

`vb_validate_stratum_methods` *Validate VegBank loader tables for new stratum methods*

Description

Performs validation checks on VegBank loader tables to ensure data integrity before upload. Validates required fields, uniqueness constraints, and referential integrity between related tables. Prints validation errors and warnings. This validation tool is a first pass at catching errors - full validation is only done at upload.

Usage

```
vb_validate_stratum_methods(stratum_methods, references = NULL)
```

Arguments

`stratum_methods`
: A data frame containing stratum methods and their associated component stratum types

`references`
: A data frame containing details about new references

Value

A named list with one element per table, each containing a logical value (TRUE if all validations passed for that table, FALSE otherwise). For example: `list(community_concepts = TRUE, community_names = FALSE, community_correlations = TRUE)`

Index