Package: inbodb (via r-universe)

August 22, 2024

Title Connect to and Retrieve Data from Databases on the INBO Server

Version 0.0.6

Description A bundle of functions to connect to and retrieve data from databases on the INBO server, with dedicated functions to query some of these databases.

License GPL-3

URL https://github.com/inbo/inbodb, https://inbo.github.io/inbodb/

BugReports https://github.com/inbo/inbodb/issues

Imports assertthat, DBI, dplyr, glue, lifecycle, methods, odbc, rlang

Suggests dbplyr, ggplot2, kableExtra, knitr, rmarkdown, sf, tidyr

VignetteBuilder knitr

Config/checklist/communities inbo

Config/checklist/keywords sql; databases; queries

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

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

RemoteUrl https://github.com/inbo/inbodb

RemoteRef HEAD

RemoteSha ced4643b25deb66c08e1d47e4631fd65058c6350

Contents

`connect_inbo_dbase` *Connect to an INBO database*

Description

Connects to an INBO database by simply providing the database's name as an argument. The function can only be used from within the INBO network.

Usage

```
connect_inbo_dbase(database_name, autoconvert_utf8 = TRUE)
```

Arguments

`database_name`	char Name of the INBO database you want to connect
`autoconvert_utf8`	Should the encoding of the tables that are retrieved from the database be adapted to ensure correct presentation? Defaults to TRUE.

Details

For more information, refer to this tutorial.

Value

odbc connection

Author(s)

Stijn Van Hoey <stijnvanhoey@gmail.com>

Els Lommelen <els.lommelen@inbo.be>

Examples

```
## Not run:
connection <- connect_inbo_dbase("D0021_00_userFlora")
connection <- connect_inbo_dbase("W0003_00_Lims")

## End(Not run)
```

---

dbDisconnect,OdbcConnection-method

Close database connection

Description

This method is an adaptation to the INBO databases from the eponymous function in the odbc package and is an implementation of the method dbDisconnect defined in the DBI package.

Usage

```
## S4 method for signature 'OdbcConnection'
dbDisconnect(conn, ...)
```

Arguments

conn	A DBIConnection object, as returned by dbConnect().
...	Other parameters passed on to methods.

---

dbFetch,OdbcResult-method

Fetch query result from database

Description

This method is an adaptation from the eponymous function in the odbc package and is an implementation of the method dbFetch defined in the DBI package. Additional to the odbc package, it replaces a cryptic error message by an informative error message.

Usage

```
## S4 method for signature 'OdbcResult'
dbFetch(res, n = -1, ...)
```

Arguments

res	An object inheriting from DBIResult, created by dbSendQuery().
n	maximum number of records to retrieve per fetch. Use n = -1 or n = Inf to retrieve all pending records. Some implementations may recognize other special values.
...	Other arguments passed on to methods.

---

get_florabank_observations

Get all validated observations for one or more taxa from the florabank database

---

Description

This function takes as input a character vector with one or more names of species either as scientific names and/or Dutch names. By default (fixed = FALSE), partial matching will be used (the names are prepended and appended with %). The function queries the florabank, and returns a dataframe with observation level information about the matching taxa.

Usage

```
get_florabank_observations(connection, names, fixed = FALSE, collect = FALSE)
```

Arguments

connection	A connection to the florabank database. See the example section for how to connect and disconnect to the database.
names	Default missing. A character vector with scientific names and/or Dutch names. If fixed = TRUE, character strings are matched exactly and scientific names must include authorship in order to match.
fixed	Logical. If TRUE, names is to be matched as is (no partial matching) .
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A dataframe with the following variables: NaamNederlands, NaamWetenschappelijk, Bron, BeginDatum, EindDatum, hok, Toponiem, CommentaarTaxon, CommentaarHabitat, WaarnemingID, X_waarneming, Y_waarneming, X_meting, Y_meting

See Also

Other florabank: get_florabank_taxon_ifbl_year(), get_florabank_traits()

Examples

```
## Not run:
# code can only be run if a connection to the database is possible
library(inbodb)
# connect to florabank
db_connectie <- connect_inbo_dbase("D0021_00_userFlora")

# query and collect the data using scientific name
succprat1 <- get_florabank_observations(db_connectie,
names = 'Succisa pratensis Moench', collect = TRUE)

# the same species but using Dutch name
succprat2 <- get_florabank_observations(db_connectie,
names = 'Blauwe knoop', collect = TRUE)

# providing both a Dutch name and scientific name will not duplicate records
# if they are the same species
succprat3 <- get_florabank_observations(db_connectie,
names = c("Succisa pratensis Moench", "Blauwe knoop"), collect = TRUE)

all.equal(succprat1, succprat2)
all.equal(succprat1, succprat3)

# passing dutch names and scientific names for different species
# is possible (records for each species is returned)
myspecies1 <- get_florabank_observations(db_connectie,
names = c('Succisa pratensis Moench', 'Gevlekte orchis'), collect = TRUE)

# passing multiple dutch names
myspecies2 <- get_florabank_observations(db_connectie,
names = c('Gevlekte orchis', 'Blauwe knoop'),
collect = TRUE)

all.equal(myspecies1, myspecies2)

# using default for collect will return a lazy query
# fixed = TRUE for exact matches only
myspecies3 <- get_florabank_observations(db_connectie,
names = c('Succisa pratensis Moench', 'Gevlekte orchis'),
fixed = TRUE)

# to collect the data for a lazy query you can also use the collect()
# function:
myspecies3 <- dplyr::collect(myspecies3)

# disconnect from florabank
dbDisconnect(db_connectie)

## End(Not run)
```

`get_florabank_taxon_ifbl_year` *Get unique combinations of taxon, IFBL-square and year.*

Description

This functions queries all validated observations of the florabank database and returns unique combinations of taxon, IFBL-square and year. Either a 1 km by 1 km or a 4 km x 4 km resolution can be chosen and a begin year can be set. Observations of taxa at genus level or higher are excluded. The taxonomic group can be chosen.

Usage

```
get_florabank_taxon_ifbl_year(
  connection,
  starting_year = 2010,
  ifbl_resolution = c("1km-by-1km", "4km-by-4km"),
 taxongroup = c("Vaatplanten", "Mossen", "Lichenen (korstmossen)", "Kranswieren"),
  collect = FALSE
)
```

Arguments

`connection`	A connection to the florabank database. See the example section for how to connect and disconnect to the database.
`starting_year`	Filter for observations that start from this year onwards. Default is 2010.
`ifbl_resolution`	The requested spatial resolution can be either 1km-by-1km IFBL squares or 4km-by-4km. Default is 1km-by-1km.
`taxongroup`	Choose for which taxonomic group you want the unique combinations. One of "Vaatplanten" (the default), "Mossen", "Korstmossen" or "Kranswieren".
`collect`	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A dataframe with one line for each combination of taxon, IFBL-square (either at 1 km x 1 km or 4 km x 4 km resolution) and year. In case the resolution is 1 km x 1 km, a variable `ifbl_4by4` gives the corresponding `ifbl_4by4` identifier within which the `ifbl_1by1` square is located. In case the resolution is 4 km x 4 km, the variable `ifbl_squares` is a concatenation of all nested squares with observations for the taxon in the corresponding year. This can be nested 1 x 1 squares as well as the corresponding 4 x 4 square (the latter is the case if the original resolution of the observation is at 4 x 4 resolution). In addition, the variable `ifbl_number_squares` gives the number of unique nested squares where the taxon was observed for that year and 4 x 4 square combination.

See Also

Other florabank: get_florabank_observations(), get_florabank_traits()

Examples

```
## Not run:
library(inbodb)
# connect to florabank
db_connectie <- connect_inbo_dbase("D0021_00_userFlora")

# get records at 1 km x 1 km resolution for vascular plants from 2010
# (default) without collecting all data into memory (default).
fb_kwartier <- get_florabank_taxon_ifbl_year(db_connectie)
# to collect the data in memory set collect to TRUE or do
fb_kwartier <- collect(fb_kwartier)

# get records at 4 km x 4 km resolution starting from 2000
fb_uur <- get_florabank_taxon_ifbl_year(db_connectie, starting_year = 2000,
 ifbl_resolution = "4km-by-4km", taxongroup = "Mossen")

# disconnect from florabank
dbDisconnect(db_connectie)

## End(Not run)
```

---

get_florabank_traits *Query the florabank to get taxon trait values for (a) taxon trait(s)*

Description

This function takes as input (part of) a taxon trait name, queries the florabank and returns the taxon trait values in a tidy data format

Usage

```
get_florabank_traits(connection, trait_name, collect = FALSE)
```

Arguments

connection	A connection to the florabank database. See the example section for how to connect and disconnect to the database.
trait_name	A (part of) a trait name for which you want to get the associated taxon-specific trait values. If this is missing, the function returns an error and prints a message showing all possible trait names.
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote `tbl` object (collect = FALSE) or a `tibble` dataframe (collect = TRUE) containing the trait values for each species and for all partially matched traits. The dataframe contains the variables `TaxonID`, `TaxonAfkorting`, `TaxonWetenschappelijk`, `TaxonNederlands`, `Kenmerk`, `Code`, `Omschrijving`, `Rekenwaarde`, `Bron` and `ExtraOmschrijving`. The first four variables identify the taxon, the latter five variables relate to the taxon traits.

See Also

Other florabank: `get_florabank_observations()`, `get_florabank_taxon_ifbl_year()`

Examples

```
## Not run:
library(inbodb)
library(dplyr)
# connect to florabank
db_connectie <- connect_inbo_dbase("D0021_00_userFlora")

# get all Ellenberg values via partial matching, return as lazy query
fb_ellenberg <- get_florabank_traits(db_connectie, "llenberg")
# collect the data
fb_ellenberg <- fb_ellenberg %>% collect()
# the same can be done by using the collect parameter
fb_ellenberg <-
  get_florabank_traits(db_connectie, "llenberg", collect = TRUE)

# get all red lists via partial matching
fb_rodelijsten <- get_florabank_traits(db_connectie, "rode")

# get only the red list for vascular plant species
fb_rodelijstvaatplanten <-
  get_florabank_traits(db_connectie, "Rode lijst Vaatplanten")

#if the trait_name argument is missing, a list of possible names is printed
get_florabank_traits(db_connectie)

#disconnect from florabank
dbDisconnect(db_connectie)

## End(Not run)
```

`get_inboveg_classification`

Query classification information from INBOVEG

Description

This function queries the INBOVEG database for information on the field classification (N2000 or local vegetation type, e.g. BWK-code) of the relevé (recording) for one or more survey(s) by the name of the survey. See the examples for how to get information for all surveys.

Usage

```
get_inboveg_classification(
  connection,
  survey_name,
  classif,
  multiple = FALSE,
  collect = FALSE
)
```

Arguments

connection	dbconnection with the database 'Cydonia' on the inbo-sql07-prd server
survey_name	A character string or a character vector giving the name or names of the survey(s) for which you want to extract Classification information. If missing, all surveys are returned.
classif	A character vector giving the Classification code of the vegetation type for which you want to extract information. If missing, all classifications are returned.
multiple	If TRUE, survey_name can take a character vector with multiple survey names that must match exactly. If FALSE (the default), survey_name must be a single character string (one survey name) that can include wildcards to allow partial matches
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables Id, SurveyName, Classification-code, vegetation type / BWK or N2000-list, LocalClassification, Description of the habitat type, Cover-code, Cover in percentage.

See Also

Other inboveg: get_inboveg_header(), get_inboveg_layer_cover(), get_inboveg_layer_qualifier(), get_inboveg_ppa(), get_inboveg_qualifier(), get_inboveg_recording(), get_inboveg_relation_recording(), get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
```

```
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get a specific classification from a survey and collect the data
classif_info <- get_inboveg_classification(con,
survey_name = "MILKLIM_Heischraal2012", classif = "4010", collect = TRUE)

# get the classification from several specific surveys
classif_info <- get_inboveg_classification(con,
  survey_name = c("MILKLIM_Heischraal2012", "NICHE Vlaanderen" ),
  multiple = TRUE)

# get all surveys, all classifications,  don't collect the data
allecodes <- get_inboveg_classification(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_inboveg_header` *Query header information from INBOVEG*

Description

This function queries the INBOVEG database for header information (metadata for a vegetation-recording or relevé) for one or more surveys and the recorder type. All records, also with 'work needed' are selected. See the examples for how to get information for all surveys.

Usage

```
get_inboveg_header(
  connection,
  survey_name,
  rec_type,
  additional_variables = character(0),
  multiple = FALSE,
  collect = FALSE
)
```

Arguments

`connection`	`dbconnection` with the database 'Cydonia' on the `inbo-sql07-prd` server
`survey_name`	A character string or a character vector giving the name or names of the survey(s) for which you want to extract header information. If missing, all surveys are returned.
`rec_type`	A character vector giving the name of record type for which you want to extract header information e.g. `'Classic'`, `'Classic-emmer'`, `'Classic-ketting'`, `'BioHab'`, `'ABS'`, `'PPA'`. If missing, all recording types are returned.

additional_variables
: Default character(0). A character vector with names of additional variables to select from ivRecording table: CoordinateRefSystem, GivenLatitute, GivenLongitude, GivenLatitude2, GivenLongitude2, Pq, Homogenous

multiple
: If TRUE, survey_name can take a character vector with multiple survey names that must match exactly. If FALSE (the default), survey_name must be a single character string (one survey name) that can include wildcards to allow partial matches

collect
: If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables RecordingGivid, SurveyName, UserReference, Observer, LocationCode, Latitude, Longitude, Area (in m2), Length (in cm), Width (in cm), VagueDateType, VagueDateBegin, VagueDateEnd, SurveyId, RecTypeID, RecTypeName.

See Also

Other inboveg: get_inboveg_classification(), get_inboveg_layer_cover(), get_inboveg_layer_qualifier(), get_inboveg_ppa(), get_inboveg_qualifier(), get_inboveg_recording(), get_inboveg_relation_recording(), get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get header information from a specific survey and a specific recording type
# and collect the data
header_info <- get_inboveg_header(con, survey_name = "OudeLanden_1979",
rec_type = "Classic", collect = TRUE)

# with additional variables
header_info <- get_inboveg_header(con, survey_name = "OudeLanden_1979",
rec_type = "Classic", additional_variables = c("Pq", "Homogenous"),
collect = TRUE)
# get header information from several specific surveys
header_severalsurveys <- get_inboveg_header(con, survey_name =
c("MILKLIM_Heischraal2012", "NICHE Vlaanderen"), multiple = TRUE)

# get header information of all surveys,  don't collect the data
all_header_info <- get_inboveg_header(con)

# close the connection when done
dbDisconnect(con)
rm(con)
```

```
## End(Not run)
```

`get_inboveg_layer_cover` *Query layer information of the cover for recordings (relevé) from INBOVEG*

Description

This function queries the INBOVEG database for layer information (layer and cover) on recordings for one or more surveys.

Usage

```
get_inboveg_layer_cover(connection, survey_name, multiple = FALSE)
```

Arguments

`connection`	`dbconnection` with the database 'Cydonia' on the `inbo-sql07-prd` server
`survey_name`	A character string or a character vector, depending on multiple parameter, giving the name or names of the survey(s) for which you want to extract recordings information. If missing, all surveys are returned.
`multiple`	If TRUE, survey_name can take a character vector with multiple survey names that must match exactly. If FALSE (the default), survey_name must be a single character string (one survey name) that can include wildcards to allow partial matches

Value

A dataframe with variables `Name` (of the survey), `RecordingGivid` (unique Id), `UserReference`, `LayerCode`, `LayerDescription`, `CoverCode`, `Coverpercentage` and `Mean height` (cm)

See Also

Other inboveg: `get_inboveg_classification()`, `get_inboveg_header()`, `get_inboveg_layer_qualifier()`, `get_inboveg_ppa()`, `get_inboveg_qualifier()`, `get_inboveg_recording()`, `get_inboveg_relation_recording()`, `get_inboveg_survey()`

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get the layer information from one survey
layerinfo_heischraal2012 <- get_inboveg_layer_cover(con, survey_name =
"MILKLIM_Heischraal2012")
```

```
# get all layer qualifiers from MILKLIM surveys (partial matching)
layerinfo_milkim <- get_inboveg_layer_cover(con, survey_name = "%MILKLIM%")

# get layer qualifiers from several specific surveys
layerinfo_severalsurveys <- get_inboveg_layer_cover(con, survey_name =
c("MILKLIM_Heischraal2012", "NICHE Vlaanderen"), multiple = TRUE)

# get all layer qualifiers of all surveys
all_layerinfo <- get_inboveg_layer_cover(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_inboveg_layer_qualifier` *Query layer qualifier information of recordings (relevé) from INBOVEG*

Description

This function queries the INBOVEG database for layer qualifier information on recordings for one or more surveys.

Usage

```
get_inboveg_layer_qualifier(connection, survey_name, multiple = FALSE)
```

Arguments

`connection`	`dbconnection` with the database 'Cydonia' on the `inbo-sql07-prd` server
`survey_name`	A character string or a character vector, depending on multiple parameter, giving the name or names of the survey(s) for which you want to extract information. If missing, all surveys are returned.
`multiple`	If TRUE, survey_name can take a character vector with multiple survey names that must match exactly. If FALSE (the default), survey_name must be a single character string (one survey name) that can include wildcards to allow partial matches

Value

A dataframe with variables `Name` (of the survey), `RecordingGivid` (unique Id), `UserReference`, `LayerCode`, `LayerDescription`, `QualifierCode`, `Qualifier Description`, `Elucidation` and `NotSure` in case the qualifier is doubtful, `CoverCode` and `Cover percentage`

See Also

Other inboveg: get_inboveg_classification(), get_inboveg_header(), get_inboveg_layer_cover(), get_inboveg_ppa(), get_inboveg_qualifier(), get_inboveg_recording(), get_inboveg_relation_recording(), get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get the layer qualifiers from one survey
layerqualifiers_Gagealutea <-
    get_inboveg_layer_qualifier(con, survey_name = "GageaLutea_1980")

# get all layer qualifiers from MILKLIM surveys (partial matching)
layerqualifiers_milkim <-
  get_inboveg_layer_qualifier(con, survey_name = "%MILKLIM%")

# get layer qualifiers from several specific surveys
layerqualifiers_severalsurveys <- get_inboveg_layer_qualifier(con,
  survey_name = c("MILKLIM_Overstroming", "NICHE Vlaanderen"),
  multiple = TRUE)

# get all layer qualifiers of all surveys
alllayerqualifiers <- get_inboveg_layer_qualifier(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_inboveg_ppa` *Query PPA (point-plant distance) information from INBOVEG*

Description

This function queries the INBOVEG database for PPA-type relevé information (which species were recorded at what distance from a point location) for one or more surveys, or in combination with the unique ID (`recordingGIVID`) or user reference Wildcards in `survey_name`, `user_reference` or `recording_givid` should only be used if a character string (a length one character vector), otherwise values are assumed to match exactly.

Usage

```
get_inboveg_ppa(
  connection,
```

```
  survey_name = "%",
  user_reference = "%",
  recording_givid = "%",
  collect = FALSE
)
```

Arguments

connection	dbconnection with the database 'Cydonia' on the inbo-sql07-prd server
survey_name	A character string or a character vector, giving the name or names of the survey(s) for which you want to extract relevé information. As default (survey_name = "%") all surveys are returned.
user_reference	A character string or a character vector giving the name of a recording for which you want to extract relevé information. As default (user_reference = "%") all user-references are returned.
recording_givid	A character string or a character vector giving the unique id of a recording for which you want to extract relevé information. As default (recording_givids = "%") all recording_givids are returned.
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables SurveyName, RecordingGivid, UserReference, DateRecording, LocationCode, CoordinateRefSystem, GivenLatitude, GivenLongitude, GivenLatitude2, GivenLongitude2, MaxSearchEffortUnit, MaxSearchEffortLabel, Indirect, NotSure, LayerCode, LayerCover, OriginalName, ScientificName, TaxonGroupCode, PhenologyCode, Distance, Comment DateIdentification, RecordTypeName

See Also

Other inboveg: get_inboveg_classification(), get_inboveg_header(), get_inboveg_layer_cover(), get_inboveg_layer_qualifier(), get_inboveg_qualifier(), get_inboveg_recording(), get_inboveg_relation_r
get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get the recordings from one survey and collect the data
specifieke_survey <- get_inboveg_ppa(con, survey_name =
"LEN_sinusmaaiproject_ppa", collect = TRUE)

# get all recordings from with partial matching, don't collect
```

```
partial_match <- get_inboveg_ppa(con, survey_name = "%LEN%",
collect = FALSE)

# get recordings from several specific recordinggivid
recording_severalgivids <- get_inboveg_ppa(con,
recording_givid = c("IV2024040411243457","IV2024040411263782"),
collect = TRUE)

# get all PPA-type recordings of all surveys,  don't collect the data
all_ppa <- get_inboveg_ppa(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_inboveg_qualifier` *Query qualifier information of recordings (relevé) from INBOVEG*

Description

This function queries the INBOVEG database for qualifier information (site qualifier or management qualifier)on recordings for one or more surveys.

Usage

```
get_inboveg_qualifier(
  connection,
  survey_name,
  qualifier_type,
  multiple = FALSE
)
```

Arguments

`connection`	`dbconnection` with the database 'Cydonia' on the `inbo-sql07-prd` server
`survey_name`	A character string or a character vector, depending on multiple parameter, giving the name or names of the survey(s) for which you want to extract recordings information. If missing, all surveys are returned.
`qualifier_type`	A character vector giving the name of qualifier type for which you want to extract information e.g. `'SQ'` (site qualifier), `'MQ'` (management qualifier). If missing, all qualifier types are returned.
`multiple`	If TRUE, survey_name can take a character vector with multiple survey names that must match exactly. If FALSE (the default), survey_name must be a single character string (one survey name) that can include wildcards to allow partial matches

Value

A dataframe with variables RecordingGivid (unique Id), UserReference, Observer, QualifierType, QualifierCode, Description, 2nd QualifierCode, 2nd Description, 3rd QualifierCode, 3rd Description, Elucidation, in case qualifier is 'NotSure', ParentID, QualifierResource

See Also

Other inboveg: get_inboveg_classification(), get_inboveg_header(), get_inboveg_layer_cover(), get_inboveg_layer_qualifier(), get_inboveg_ppa(), get_inboveg_recording(), get_inboveg_relation_record
get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get the qualifiers from one survey
qualifiers_heischraal2012 <- get_inboveg_qualifier(con, survey_name =
"MILKLIM_Heischraal2012")

# get all site qualifiers (SQ) from MILKLIM surveys (partial matching)
qualifiers_milkim <- get_inboveg_qualifier(con, survey_name = "%MILKLIM%",
qualifier_type = "SQ")

# get qualifiers from several specific surveys
qualifiers_severalsurveys <- get_inboveg_qualifier(con, survey_name =
c("MILKLIM_Heischraal2012", "NICHE Vlaanderen"), multiple = TRUE)

# get all qualifiers of all surveys
allqualifiers <- get_inboveg_qualifier(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

---

get_inboveg_recording *Query recording (relevé) information from INBOVEG*

Description

This function queries the INBOVEG database for relevé information (which species were recorded in which plots and in which vegetation layers with which cover) for one or more surveys, or in combination with the unique ID (recordingGIVID) or user reference Wildcards in survey_name, user_reference or recording_givid should only be used if a character string (a length one character vector), otherwise values are assumed to match exactly.

Usage

```
get_inboveg_recording(
  connection,
  survey_name = "%",
  user_reference = "%",
  recording_givid = "%",
  collect = FALSE,
  multiple = deprecated()
)
```

Arguments

connection	dbconnection with the database 'Cydonia' on the inbo-sql07-prd server
survey_name	A character string or a character vector, giving the name or names of the survey(s) for which you want to extract relevé information. As default (survey_name = "%") all surveys are returned.
user_reference	A character string or a character vector giving the name of a recording for which you want to extract relevé information. As default (user_reference = "%") all user-references are returned.
recording_givid	A character string or a character vector giving the unique id of a recording for which you want to extract relevé information. As default (recording_givids = "%") all recording_givids are returned.
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.
multiple	Deprecated.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables RecordingGivid (unique ID), User reference, LayerCode, CoverCode, OriginalName, ScientificName, TaxonGroupCode, PhenologyCode, Comment, CoverageCode, PctValue (percentage coverage), RecordingScale (name of the scale of coverage)

See Also

Other inboveg: get_inboveg_classification(), get_inboveg_header(), get_inboveg_layer_cover(), get_inboveg_layer_qualifier(), get_inboveg_ppa(), get_inboveg_qualifier(), get_inboveg_relation_record
get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")
```

```
# get the recordings from one survey and collect the data
recording_heischraal2012 <- get_inboveg_recording(con, survey_name =
"MILKLIM_Heischraal2012", collect = TRUE)

# get all recordings from MILKLIM surveys (partial matching), don't collect
recording_milkim <- get_inboveg_recording(con, survey_name = "%MILKLIM%",
collect = FALSE)

# get recordings from several specific surveys
recording_severalsurveys <- get_inboveg_recording(con, survey_name =
c("MILKLIM_Heischraal2012", "NICHE Vlaanderen"),
collect = TRUE)

# get recordings from several specific recordinggivid
recording_severalgivids <- get_inboveg_recording(con,
recording_givid = c("IV2012081609450300","IV2012081610204607"),
collect = TRUE)

# get all recordings of all surveys,  don't collect the data
allrecordings <- get_inboveg_recording(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_inboveg_relation_recording` *Query relation (Parent - Child) information of recordings (relevé) from INBOVEG*

Description

This function queries the INBOVEG database for relation information on recordings for one or more surveys based on Parent (classic-chain/bucket) and Child (classic) relationship.

Usage

```
get_inboveg_relation_recording(
  connection,
  survey_name,
  multiple = FALSE,
  collect = FALSE
)
```

Arguments

`connection`	`dbconnection` with the database 'Cydonia' on the `inbo-sql07-prd` server

survey_name	A character string or a character vector, depending on multiple parameter, giving the name or names of the survey(s) for which you want to extract recordings information. If missing, all surveys are returned.
multiple	If TRUE, survey_name can take a character vector with multiple survey names that must match exactly. If FALSE (the default), survey_name must be a single character string (one survey name) that can include wildcards to allow partial matches
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A dataframe with variables RecordingId, Child_GIVID (unique RecordingGIVID), Child_UserRef (UserReference), ParentId (RecordingId), Parent_GIVID (unique RecordingGIVID) and Parent_UserRef (UserReference)

See Also

Other inboveg: get_inboveg_classification(), get_inboveg_header(), get_inboveg_layer_cover(), get_inboveg_layer_qualifier(), get_inboveg_ppa(), get_inboveg_qualifier(), get_inboveg_recording(), get_inboveg_survey()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get the Parent-Child-relations from one survey
relations_N2000meetnet_Grasland <- get_inboveg_relation_recording(con,
    survey_name = "N2000meetnet_Grasland")

# get all Parent-Child-relations from N2000meetnet surveys (partial matching)
relations_N2000meetnet <-
    get_inboveg_relation_recording(con, survey_name = "%N2000meetnet%")

# get Parent-Child-relations from several specific surveys
relations_severalsurveys <-
   get_inboveg_relation_recording(con,
   survey_name = c("DeBlankaart-1985-Beheer", "N2000meetnet_Grasland"),
   multiple = TRUE)

# get all Parent-Child-relations of all relevant surveys
allrelations <- get_inboveg_relation_recording(con)

# Close the connection when done
dbDisconnect(con)
rm(con)
```

```
## End(Not run)
```

`get_inboveg_survey` *Query survey information from INBOVEG*

Description

This function queries the INBOVEG database for survey information (metadata about surveys) for one or more survey(s) by the name of the survey. See the examples for how to get information for all surveys.

Usage

```
get_inboveg_survey(connection, survey_name, collect = FALSE)
```

Arguments

`connection`	`dbconnection` with the database 'Cydonia' on the `inbo-sql07-prd` server
`survey_name`	A character vector giving the names of the surveys for which you want to extract survey information.
`collect`	If FALSE (the default), a remote `tbl` object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote `tbl` object (collect = FALSE) or a `tibble` dataframe (collect = TRUE) with variables Id, Name, Description, Owner and Creator.

See Also

Other inboveg: `get_inboveg_classification()`, `get_inboveg_header()`, `get_inboveg_layer_cover()`, `get_inboveg_layer_qualifier()`, `get_inboveg_ppa()`, `get_inboveg_qualifier()`, `get_inboveg_recording()`, `get_inboveg_relation_recording()`

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0010_00_Cydonia")

# get information of a specific survey and collect data
survey_info <- get_inboveg_survey(con, survey_name = "OudeLanden_1979",
collect = TRUE)

# get information of all surveys and collect data
```

```
allsurveys <- get_inboveg_survey(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_meetnetten_locations` *Query monitoring scheme locations from Meetnetten*

Description

This function queries the Meetnetten database for the locations and sublocations for a specified monitoring scheme or for all monitoring schemes within a specified species group. When no monitoring scheme or species group is specified, the observations of all monitoring schemes are returned.

Usage

```
get_meetnetten_locations(connection, scheme_name = NULL, species_group = NULL)
```

Arguments

`connection`	dbconnection with the database 'S0008_00_Meetnetten' on the inbo-sql08-prd.inbo.be server
`scheme_name`	the name of the monitoring scheme for which you want to extract location data. Data from multiple schemes can be selected by providing a vector with the names of the schemes.
`species_group`	the name of the species group for which you want to extract location data. Data from multiple species groups can be selected by providing a vector with the names of the species groups.

Details

Each monitoring scheme of the species monitoring programme of Flanders Meetnetten consists of a fixed set of locations. A monitoring scheme for rare species includes all locations where the species occurs. For more common species a sample of locations is drawn and the the selected locations are included in the monitoring scheme. In some cases, the monitoring project in Meetnetten also contains locations that are not part of the sample. These locations can be counted optionally and are indicated by (`is_sample = FALSE`).

It also occurs that a location becomes inaccessible or that the target species disappears. Then, a locations can be made inactive (`is_active = FALSE`), which means that no observations can be recorded any more.

Value

When the sf package is installed, a list with two sf objects is returned:

- main_locations: the main locations of the selected monitoring schemes, with following attribute variables:
  - species_group
  - scheme: name of the monitoring scheme
  - location: name of the location
  - is_sample: whether the location belongs to the sample of locations for the monitoring scheme (see details)
  - is_active: when a location is not suited for counting any more, the location becomes inactive (is_active = FALSE)
- sublocations: the sublocations (for example the sections of a transect) for each of the selected main locations, with following attribute variables:
  - species_group
  - scheme: name of the monitoring scheme
  - location: name of the main location
  - sublocation: name of the sublocation
  - is_active: whether the sublocation is counted or not

When the sf package is not installed, a list with two tibble objects is returned, with the same attribute variables as above and an additional variable geom that contains the geometry information in wkt (well-known text) format.

Not all main locations are subdivided in sublocations. So in some cases the sublocations object is empty.

See Also

Other meetnetten: get_meetnetten_observations(), get_meetnetten_schemes(), get_meetnetten_visits()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("S0008_00_Meetnetten")

# get locations for a specific monitoring scheme
locations_heivlinder <- get_meetnetten_locations(con,
                                                 scheme_name = "Heivlinder")

locations_heivlinder$main_locations
locations_heivlinder$sublocations

# get locations for a specific species_group
locations_dragonflies <- get_meetnetten_locations(con,
                                                  species_group = "libellen")

locations_dragonflies$main_locations
```

```
locations_dragonflies$sublocations

# Close the connection when done
dbDisconnect(con)
rm(con)
rm(locations_heivlinder)
rm(locations_dragonflies)

## End(Not run)
```

`get_meetnetten_observations` *Query observation data from Meetnetten*

Description

This function queries the Meetnetten database for observation data (standardized counts) for a specified monitoring scheme or for all monitoring schemes within a specified species group. When no monitoring scheme or species group is specified, the observations of all monitoring schemes are returned.

Usage

```
get_meetnetten_observations(
  connection,
  scheme_name = NULL,
  species_group = NULL,
  collect = FALSE
)
```

Arguments

`connection`	dbconnection with the database 'S0008_00_Meetnetten' on the inbo-sql08-prd.inbo.be server
`scheme_name`	the name of the monitoring scheme for which you want to extract visit data. Data from multiple schemes can be selected by providing a vector with the names of the schemes.
`species_group`	the name of the species group for which you want to extract visit data. Data from multiple species groups can be selected by providing a vector with the names of the species groups.
`collect`	If `FALSE` (the default), a remote `tbl` object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If `TRUE` the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Details

The species monitoring programme of Flanders (Meetnetten) consists of a series of monitoring schemes in which one or more target species are counted based on a specific protocol. Optionally, other species, that can be counted using the same protocol, can be recorded as well. When `checklist_complete = TRUE`, all secondary species were counted, and we can assume that the secondary species that were not recorded are absent.

Depending on the protocol, counting has to be done at the location or the sublocation level. Sublocations are, for example, different sections of a transect. For some monitoring schemes, it is necessary to record several count subevents at the location level. This is, for example, the case for the crested newt fyke count protocol, where two fykes are used per location and the counts are recorded per fyke. For every count subevent a unique `sample_id` is created.

The protocol of a monitoring scheme also defines for which combinations of sex, life stage, and activity type the counts have to be recorded. For example, for the crested newt fyke counts the number of female adults, male adults and juveniles (sex undefined) are counted. Another example: in the alcon blue monitoring scheme only the number of eggs are counted.

It is also important to know that counts can be recorded in the Meetnetten website or by using the Meetnetten app. When using the Meetnetten app, the GPS coordinates of all observations are recorded and the observations are assigned to a location or sublocation based on the coordinates. For example, when you record a butterfly transect count in the website, you will enter the total number of individuals per species for each section (the sublocation) of the transect. When you use the app, you can record the position of every individual separately in the Meetnetten database. So when you want to know the total number of individuals per section, you will have to aggregate the data.

To conclude, it is important to understand how the data is organised for a certain monitoring scheme, before you start analysing the data. For more details on the monitoring schemes we refer to Maes et al. (2023)

Value

A remote `tbl` object (`collect = FALSE`) or a `tibble` dataframe (`collect = TRUE`) with following variables:

- `species_group`
- `scheme`
- `protocol`: the protocol used
- `visit_id`: unique id for a count event
- `start_date`:date of the observation
- `location`: the name of the location
- `sublocation`: the name of the sublocation
- `not_counted`: `TRUE` when the sublocation is not counted
- `sample_id`: unique id for a count subevent (see details)
- `target_species`: `TRUE` when the observed species is the target species, `FALSE` when the observed species is a secondary species (another species than the target species that can be counted with the same protocol, see details)

- `checklist_complete`: whether all secondary species, defined in the monitoring scheme, are counted
- `name_nl`: Dutch name of the observed species
- `scientific_name`: scientific name of the observed species
- `sex`: M (male), F (female), U (undefined)
- `activity`: activity of the observed species
- `life_stage`: live stage of the observed species
- `count`: number of individuals counted
- `count_type`: most of the time the number of individuals are counted (`count_type = exact count`), however for some monitoring schemes different type of counts are performed. Check the protocol for more information when this is the case.
- `notes`: notes of the observed
- `x` and `y`: when the Meetnetten-app is used, GPS coordinates (longitude and latitude, `crs = WGS84`) of each observation is recorded

References

- Maes D, Piesschaert F, Ledegen H, Van De Poel S, Adriaens T, Anselin A, Belpaire C, Breine J, Brosens D, Brys R, De Bruyn L, Decleer K, De Knijf G, Devos K, Driessens G, Feys S, Gouwy J, Gyselings R, Herremans M, Jacobs I, Lewylle I, Leyssen A, Louette G, Onkelinx T, Packet J, Provoost S, Quataert P, Ruyts S, Scheppers T, Speybroeck J, Steeman R, Stienen E, Thomaes A, Van Den Berge K, Van Keer K, Van Landuyt W, Van Thuyne G, Veraghtert W, Verbelen D, Verbeylen G, Vermeersch G, Westra T, Pollet M (2023). Monitoring schemes for species of conservation concern in Flanders (northern Belgium). An overview of established schemes and the design of an additional monitoring scheme. Reports of the Research Institute for Nature and Forest (INBO) 2023 (15). Research Institute for Nature and Forest (INBO), Brussels. doi:10.21436/inbor.93332112.

See Also

Other meetnetten: `get_meetnetten_locations()`, `get_meetnetten_schemes()`, `get_meetnetten_visits()`

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("S0008_00_Meetnetten")

# get observations for a specific monitoring scheme and collect data
get_meetnetten_observations(con, scheme_name = "Boomkikker", collect = TRUE)

# get observations for a specific species_group and collect data
get_meetnetten_observations(con, species_group = "libellen", collect = TRUE)

# get observations for all species and do not collect data
observations_all <- get_meetnetten_observations(con)

# Close the connection when done
```

```
dbDisconnect(con)
rm(con)
rm(observations_all)

## End(Not run)
```

`get_meetnetten_schemes` *Overview of monitoring schemes in the Meetnetten database*

Description

This function queries the Meetnetten database to give an overview of monitoring schemes that are included.

Usage

```
get_meetnetten_schemes(connection)
```

Arguments

`connection`	dbconnection with the database 'S0008_00_Meetnetten' on the inbo-sql08-prd.inbo.be server.

Details

The species monitoring programme of Flanders (Meetnetten) consists of a series of monitoring schemes. In each monitoring scheme one or more target species are counted based on a specific protocol. For more details we refer to Maes et al. (2023)

Value

A tibble dataframe with variables species_group, scheme and protocol.

References

- Maes D, Piesschaert F, Ledegen H, Van De Poel S, Adriaens T, Anselin A, Belpaire C, Breine J, Brosens D, Brys R, De Bruyn L, Decleer K, De Knijf G, Devos K, Driessens G, Feys S, Gouwy J, Gyselings R, Herremans M, Jacobs I, Lewylle I, Leyssen A, Louette G, Onkelinx T, Packet J, Provoost S, Quataert P, Ruyts S, Scheppers T, Speybroeck J, Steeman R, Stienen E, Thomaes A, Van Den Berge K, Van Keer K, Van Landuyt W, Van Thuyne G, Veraghtert W, Verbelen D, Verbeylen G, Vermeersch G, Westra T, Pollet M (2023). Monitoring schemes for species of conservation concern in Flanders (northern Belgium). An overview of established schemes and the design of an additional monitoring scheme. Reports of the Research Institute for Nature and Forest (INBO) 2023 (15). Research Institute for Nature and Forest (INBO), Brussels. doi:10.21436/inbor.93332112.

See Also

Other meetnetten: `get_meetnetten_locations()`, `get_meetnetten_observations()`, `get_meetnetten_visits()`

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("S0008_00_Meetnetten")

# get overview of monitoring schemes in meetnetten database
meetnetten_schemes <- get_meetnetten_schemes(con)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

---

get_meetnetten_visits *Query visit data from Meetnetten*

Description

This function queries the Meetnetten database for visit data (data about a counting event) for a specified monitoring scheme or for all monitoring schemes within a specified species group. When no monitoring scheme or species group is specified, the visits of all monitoring schemes are returned.

Usage

```
get_meetnetten_visits(
  connection,
  scheme_name = NULL,
  species_group = NULL,
  collect = FALSE
)
```

Arguments

`connection`	dbconnection with the database 'S0008_00_Meetnetten' on the inbo-sql08-prd.inbo.be server.
`scheme_name`	the name of the monitoring scheme for which you want to extract visit data.
`species_group`	the name of the species group for which you want to extract visit data.
`collect`	If `FALSE` (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If `TRUE` the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with following variables:

- species_group
- scheme: the name of the monitoring scheme
- protocol: the protocol used
- location: the name of the location
- visit_id: unique id for a count event
- validation_status: validation status of the visit (visits that are validated and not approved are not provided)
  - 10: visit not validated
  - 100: visit validated and approved
- start_date: the start date of the visit
- start_time: the start time of the visit
- end_date: the end date of the visit
- end_time: the end time of the visit
- date_created: the date at which the data was imported in the database
- visit_status: the status of the visit (determined by the observer) using following categories:
  - conform protocol: the protocol was applied
  - weersomstandigheden waren ongunstig: weather conditions were unfavourable
  - telmethode uit handleiding niet gevolgd: the protocol was not applied
  - geen veldwerk mogelijk - locatie ontoegankelijk: counting was not possible because the location is inaccessible
  - geen veldwerk mogelijk - locatie is ongeschikt voor de soort: counting was not possible because the location is not suitable for the species
- for_analysis: whether the data is suited for analysis (determined by the validator)
- for_targets: every year targets are set in terms of the number of locations that have to be counted per monitoring scheme; when for_targets = TRUE the visit contributes to these targets
- notes: notes by the observer

See Also

Other meetnetten: get_meetnetten_locations(), get_meetnetten_observations(), get_meetnetten_schemes()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("S0008_00_Meetnetten")

# get visits for a specific monitoring scheme and collect data
```

```
get_meetnetten_visits(con, scheme_name = "Boomkikker", collect = TRUE)

# get visits for a specific species_group and collect data
get_meetnetten_visits(con, species_group = "libellen", collect = TRUE)

# get visits for all species and do not collect data
visits_all <- get_meetnetten_visits(con)

# Close the connection when done
dbDisconnect(con)
rm(con)
rm(visits_all)

## End(Not run)
```

`get_taxonlijsten_features` *Query to extract Taxonlist features from* `D0156_00_Taxonlijsten`

Description

This function queries `D0156_00_Taxonlijsten` and gives an overview of all the features associated with a `TaxonlijstVersie` (a red list status or an annex of the Habitat Directive are examples of a feature). This is an auxiliary function to check the accepted values (KenmerkwaardeCodes) of the feature parameter in the core function `get_taxonlijsten_items`

Usage

```
get_taxonlijsten_features(
  connection,
  list = "%",
  version = c("latest", "old", "all"),
  collect = FALSE
)
```

Arguments

`connection`	dbconnection with the database `D0156_00_Taxonlijsten` on the inbo-sql07-prd server
`list`	name of the taxonlist that you want to retrieve. Wildcards % are allowed. Case insensitive.
`version`	A choice ('latest', 'old', 'all'). If 'latest' (the default) only the most recent version is returned. If 'old' all but the most recent version is returned. If 'all' all versions are returned.
`collect`	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables Taxonlijst, Publicatiejaar, Version, Kenmerkcode, KenmerkBeschrijving, KenmerkwaardeCode, KenmerkwaardeBeschrijving

See Also

Other taxonlijsten: `get_taxonlijsten_items()`, `get_taxonlijsten_lists()`

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0156_00_Taxonlijsten")

# get features of all versions of the 'Rode lijst van de Dagvlinders'
get_taxonlijsten_features(con, version = 'all', list = '%rode%dagvlinders%'
, collect = TRUE)

# get features of Habitattypical fauna
get_taxonlijsten_features(con, list = '%Habitattyp%fauna%')

# use function with default values (all features of recent versions)
get_taxonlijsten_features(con)

# note that function also returns taxonlists without features
get_taxonlijsten_features(con, list = '%SBP%')

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_taxonlijsten_items` *Query to extract the taxa on a taxonlist from* `D0156_00_Taxonlijsten`

Description

This function queries `D0156_00_Taxonlijsten` and gives an overview of the taxa that are on a given taxon list version. The interpreted taxa are given by default, but it is possible to add taxa as they were originally published. The taxa of the latest list version are shown unless specified otherwise.

Usage

```
get_taxonlijsten_items(
  connection,
  list = "%",
  taxon = "%",
  feature = "%",
  version = c("latest", "old", "all"),
  original = FALSE,
  collect = FALSE
)
```

Arguments

connection	dbconnection with the database D0156_00_Taxonlijsten on the inbo-sql07-prd server
list	name of the taxonlist that you want to retrieve. Wildcards % are allowed. Case insensitive.
taxon	name of the taxon you want to retrieve. Scientific and vernacular (Dutch) names are allowed. Wildcards % are allowed. Case insensitive.
feature	name of the list feature (actually feature code) you want to retrieve. Wildcards % are allowed. Case insensitive.
version	A choice ('latest', 'old', 'all'). If 'latest' (the default) only the most recent version is returned. If 'old' all but the most recent version is returned. If 'all' all versions are returned.
original	If FALSE (the default), the function will only retrieve the interpreted taxa. If TRUE, columns with the original taxa will be added to the output. For example, if the originally published taxon on a taxonlist is 'Cicindela spec.', the interpretation will exist of all relevant Cicindela species
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables Lijst, Publicatiejaar, LaatsteVersie, Taxongroep, Naamwet_interpretatie, Auteur, NaamNed_interpretatie, Kenmerk, KenmerkwaardeCode, Kenmerkwaarde and extra variables Taxongroep_origineel, Naamwet_origineel, Naamned_origineel when requested (original = TRUE)

See Also

Other taxonlijsten: get_taxonlijsten_features(), get_taxonlijsten_lists()

Examples

```
## Not run:
library(inbodb)
library(tidyverse)
con <- connect_inbo_dbase("D0156_00_Taxonlijsten")

# Get all taxa from list 'Jachtdecreet'
get_taxonlijsten_items(con, list =  'Jachtdecreet', collect = TRUE)

# Get all taxa on category 2 of 'Soortenbesluit'
get_taxonlijsten_items(con, list =  'soortenbesluit', feature = 'cat2')

# Get all taxonlist that include 'Gentiaanblauwtje'
get_taxonlijsten_items(con, taxon = 'Gentiaanblauwtje', collect = TRUE)

# Get all taxa with red list status CR (critically endangered)
get_taxonlijsten_items(con, feature = 'CR')

# Get original and interpreted Cicindela taxa from list 'Soortenbesluit'
get_taxonlijsten_items(con, list = 'Soortenbesluit', taxon = '%Cicindela%'
, original = TRUE) %>%
select('Naamwet_origineel', 'NaamNed_origineel', 'Naamwet_interpretatie'
, 'NaamNed_interpretatie')

# Compare red list status on multiple listversions
get_taxonlijsten_items(con, version = 'all'
, list = 'rode lijst van de dagvlinders') %>%
select('Lijst', 'Publicatiejaar', 'Naamwet_interpretatie'
, 'NaamNed_interpretatie', 'KenmerkwaardeCode') %>%
pivot_wider(names_from = Publicatiejaar, values_from = KenmerkwaardeCode)

# Close the connection when done
dbDisconnect(con)
rm(con)

## End(Not run)
```

`get_taxonlijsten_lists` *Query to extract Taxonlijsten from* `D0156_00_Taxonlijsten`

Description

This function queries `D0156_00_Taxonlijsten` and gives an overview of all the taxon lists and list versions currently available in the database. Only the latest version is shown unless specified otherwise

Usage

```
get_taxonlijsten_lists(
  connection,
  list = "%",
  version = c("latest", "old", "all"),
  collect = FALSE
)
```

Arguments

connection	dbconnection with the database D0156_00_Taxonlijsten on the inbo-sql07-prd server
list	name of the taxonlist that you want to retrieve. Wildcards % are allowed. Case insensitive.
version	A choice ('latest', 'old', 'all'). If 'latest' (the default) only the most recent version is returned. If 'old' all but the most recent version is returned. If 'all' all versions are returned.
collect	If FALSE (the default), a remote tbl object is returned. This is like a reference to the result of the query but the full result of the query is not brought into memory. If TRUE the full result of the query is collected (fetched) from the database and brought into memory of the working environment.

Value

A remote tbl object (collect = FALSE) or a tibble dataframe (collect = TRUE) with variables TaxonlijstType, TaxonlijstCode, Taxonlijst, Publicatiejaar, Version, ReferentieURL, Criteria, Validering, Vaststelling.

See Also

Other taxonlijsten: get_taxonlijsten_features(), get_taxonlijsten_items()

Examples

```
## Not run:
library(inbodb)
con <- connect_inbo_dbase("D0156_00_Taxonlijsten")

# get the most recent version of the 'Rode lijst van de Dagvlinders'
get_taxonlijsten_lists(con, version = 'latest', list =
'%rode%dagvlinders%', collect = FALSE)

# get all recent red lists
get_taxonlijsten_lists(con, list = '%rode lijst%')

# get all taxonlist versions in the database
get_taxonlijsten_lists(con, version = 'all', collect = TRUE)

# use function with default values (only most recent versions)
```

```
get_taxonlijsten_lists(con)

# status of red lists
rl <- get_taxonlijsten_lists(con, list = '%rode lijst%')
select(rl,"Taxonlijst", "PublicatieJaar", "Criteria", "Validering",
"Vaststelling")

# Close the connection when done
dbDisconnect(con)
rm(con, rl)

## End(Not run)
```

Index