Package: austraits (via r-universe)

December 8, 2024

Title Helpful functions to access the AusTraits database and wrangle data from other traits.build databases

Version 3.0.1

Description `austraits` allow users to **access, explore and wrangle data** from traits.build relational databases. It is also an R interface to AusTraits, the Australian plant trait database. This package contains functions for joining data from various tables, filtering to specific records, combining multiple databases and visualising the distribution of the data. We expect this package will assist users in working with `traits.build` databases.

License MIT + file LICENSE

Encoding UTF-8

Language en

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 4.0.0), RefManageR

Imports dplyr, tidyr, rlang, purrr, tidyselect, stringr, stats, jsonlite, utils, magrittr, janitor, lifecycle, tibble, cli

Suggests ggplot2, ggpointdensity, ggbeeswarm (>= 0.7.1), gridExtra, readr, scales, forcats, viridis, lubridate, knitr, rmarkdown, testthat (>= 3.0.0), markdown

VignetteBuilder knitr

Config/testthat/edition 3

URL https://traitecoevo.github.io/austraits/

BugReports https://github.com/traitecoevo/austraits/issues

Config/pak/sysreqs libicu-dev libxml2-dev libssl-dev

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

RemoteUrl https://github.com/traitecoevo/austraits

RemoteRef HEAD

RemoteSha 9935b90f9c8c70845b97521dd86fda2a08f501a7

Contents

`as_wide_table` *Create a single wide table from a traits.build data object*

Description

Create a single wide table from a traits.build data object

Usage

```
as_wide_table(database)
```

Arguments

`database`	traits.build database (list object)

Value

A single wide table with collapsed contexts and locations text and with some cols renamed for alignment with other resources

Examples

```
## Not run:
austraits %>% as_wide_table()

## End(Not run)
```

`bind_databases` *Bind multiple traits.build data objects into a single data object*

Description

`bind_databases` binds all the listed studies into a single traits.build database object as a large list.

Usage

```
bind_databases(..., databases = list(...))
```

Arguments

`...`	Arguments passed to other functions
`databases`	List of traits.build databases to be bond together

Value

Compiled database as a single large list

`bind_trait_values` *Bind trait values*

Description

This function condenses data for studies that have multiple observations for a given trait into a single row. This function concatenates multiple values into a single cell

Usage

```
bind_trait_values(trait_data)
```

Arguments

`trait_data`	the traits table in a traits.build database – see example

Value

tibble that is condensed down where multiple observations in value, value_type and replicates are collapsed down and separated by '–'

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Examples

```
## Not run:
traits <- austraits$traits %>%
dplyr::filter(dataset_id == "ABRS_1981")
traits
traits_bind <- bind_trait_values(traits)

## End(Not run)
```

`convert_df_to_list` *Convert dataframe to list*

Description

Convert a dataframe to a named list, useful when converting a datafreme a to yaml.

Usage

```
convert_df_to_list(df)
```

Arguments

`df`	A dataframe

Value

A (yaml) list

Examples

```
convert_df_to_list(dplyr::starwars)
```

`convert_list_to_df1` *Convert list with single entries to dataframe*

Description

Convert a list with a single level of entries to a dataframe, useful when converting a yaml into a dataframe.

Usage

```
convert_list_to_df1(my_list)
```

Arguments

`my_list`	A list with single entries

Value

A tibble with two columns

Examples

```
## Not run:
convert_list_to_df1(as.list(dplyr::starwars)[2])

## End(Not run)
```

convert_list_to_df2 *Convert list of lists to dataframe*

Description

Convert a list of lists to a dataframe, useful when converting a multi-level yaml into a dataframe. Function required that every list have same named elements.

Usage

```
convert_list_to_df2(my_list, as_character = TRUE, on_empty = NA)
```

Arguments

`my_list`	A list of lists to dataframe
`as_character`	A logical value, indicating whether the values are read as character
`on_empty`	Value to return if my_list is NULL, NA or is length == 0, default = NA

Value

tibble

Examples

```
demo_list1 <- list(word1 = "this", word2 = "is", word3 = "an", word4 = "example", word5 = "list")
demo_list2 <- list(word1 = "and", word2 = "a", word3 = "second", word4 = "list", word5 = "also")
combined_list <- list(demo_list1, demo_list2)
convert_list_to_df2(combined_list)
```

extract_data *Extract data from traits.build database*

Description

Function to extract data from a traits.build database based on any value(s) from any column in the traits, locations, contexts, methods, taxa, taxonomic_updates, and contributors tables. The output a traits.build formatted database with all tables subset based on the specified table, column (variable) and column value.

Usage

```
extract_data(database, table = NA, col, col_value)
```

Arguments

`database`	traits.build database (list object)
`table`	Table within a traits.build database
`col`	Column name within the specified table.
`col_value`	Value (of column, from with a table) that is used to subset database. This can be a single value or a vector. It includes partial string matches.

Value

subset traits.build database

Examples

```
## Not run:
extract_data(database = traits.build_database, table = "traits",
col = "trait_name", col_value = "leaf_area")

## End(Not run)
```

`extract_dataset` *Extract all data for a particular dataset*

Description

Function to subset all data associated with a particular dataset from a traits.build relational database.

Usage

```
extract_dataset(database, dataset_id)
```

Arguments

`database`	traits.build database (list object)
`dataset_id`	character string that matches a `dataset_id` in the database

Details

`extract_dataset` has been developed to extract data for specific datasets from databases built using the traits.build workflow. Learn more at: https://github.com/traitecoevo/traits.build & https://github.com/traitecoevo/traits.build-book

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see https://github.com/traitecoevo/austraits for how to install old versions of the package or download a newer version of the database.

Value

List of tibbles containing all traits.build data and metadata for the specified dataset(s).

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Examples

```
## Not run:
extract_dataset(database, "Falster_2003")

## End(Not run)
```

`extract_taxa` *Extract all data for specific taxa*

Description

Function to subset of all data associated with a particular taxon from a traits.build relational database.

Usage

```
extract_taxa(database, family = NULL, genus = NULL, taxon_name = NULL)
```

Arguments

`database`	traits.build database (list object)
`family`	character string of family or families
`genus`	character string of genus or genera
`taxon_name`	character string of taxon name(s)

Details

`extract_taxa` has been developed to extract data for specific taxa from databases built using the traits.build workflow. Learn more at: https://github.com/traitecoevo/traits.build & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/database` for how to install old versions of the package or download a newer version of the database.

Value

List of tibbles containing all traits.build data and metadata for the specified taxa.

Author(s)

Fonti Kar - f.kar@unsw.edu.au

Examples

```
## Not run:
extract_taxa(database = austraits, family = "Proteaceae")
extract_taxa(database = austraits, genus = "Acacia")

## End(Not run)
```

`extract_trait` *Extract all data for specific traits*

Description

Function to subset all data associated with a particular trait from a traits.build relational database.

Usage

```
extract_trait(database, trait_names, taxon_names)
```

Arguments

`database`	traits.build database (list object)
`trait_names`	character string of trait(s) for which data will be extracted
`taxon_names`	optional argument, specifying taxa for which data will be extracted

Details

`extract_trait` has been developed to extract data for specific traits from databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

List of tibbles containing all traits.build data and metadata for the specified trait(s).

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Examples

```
## Not run:
extract_trait(database = austraits, trait_names = "wood_density", taxon_names = "Acacia celsa")

## End(Not run)
```

`flatten_database` *Create combined traits.build table*

Description

Create a single database output that merges together the information in all relational tables within a traits.build database. Trait measurements are still output in long format (1 row per trait value), but all measurement-related metadata (methods, location properties, context properties, contributors) are now included as additional columns in a single table.

Usage

```
flatten_database(database, format, vars, include_description)
```

Arguments

`database`	traits.build database (list object)
`format`	A parameter for the locations, contexts and data contributors tables specifying how data are packed. All three can be formatted as a single compacted column(s) will have a human readable column ("single_column_pretty") or using json ("single_column_json") syntax. For location properties or context properties there is also the option to add each `location_property` or `context_property` to the traits table as its own column ("many_columns"); the contributors column defaults to "single_column_pretty" when this option is selected.
`vars`	List specifying which columns or properties to include from each table. The detail is for all columns/properties to be included.
`include_description`	A logical indicating whether to include (TRUE) or omit (FALSE) the context_property descriptions; defaults to TRUE.

Value

A table combining information in 7 traits.build relational tables: traits, locations, contexts, methods, taxa, taxonomic_updates, and contributors

`get_compiled_by_traits.build` *Retrieve compiled by information from metadata table*

Description

Retrieve compiled by information from metadata table

Usage

```
get_compiled_by_traits.build(database)
```

Arguments

`database`	traits.build database

Value

logical, TRUE indicating version traits table came from traits.build version > 1.0

`get_traits_table` *Retrieve traits table if user passes traits.build object.*

Description

Retrieve traits table if user passes traits.build object.

Usage

```
get_traits_table(database)
```

Arguments

`database`	traits.build database or traits table in a traits.build database

get_versions *Print out AusTraits versions*

Description

Print out AusTraits versions

Usage

```
get_versions(path = "data/austraits", update = TRUE)
```

Arguments

path	A file path where AusTraits was previously downloaded
update	Would you like the versions json be updated in case of new releases?

Value

A tibble containing version numbers and doi which can be used in load_austraits()

Examples

```
## Not run:
austraits <- load_austraits(version = "3.0.2", path = "data/austraits")

## End(Not run)
```

get_version_latest *Retrieve the latest version of AusTraits*

Description

Retrieve the latest version of AusTraits

Usage

```
get_version_latest(path = "data/austraits", update = TRUE)
```

Arguments

path	file path to where AusTraits will be downloaded
update	if TRUE, AusTraits versions .json will be re-downloaded

Value

character string of latest version

`join_context_properties` *Joining context properties to traits table*

Description

Function to merge metadata from the contexts table of a traits.build database into the core traits table.

Usage

```
join_context_properties(
  database,
  format = "single_column_pretty",
  vars = "all",
  include_description = TRUE
)
```

Arguments

`database`	traits.build database (list object)
`format`	Specifies whether metadata from the contexts is output in a human readable format ("single_column_pretty"; default), with each context property added as a separate column ("many_columns") or using json syntax ("single_column_json").
`vars`	Location properties for which data is to be appended to the traits table, defaulting to all context properties (vars = "all").
`include_description`	A logical indicating whether to include (TRUE) or omit (FALSE) the context_property descriptions.

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but context properties from the contexts table appended to the traits table.

Examples

```
## Not run:
(database %>% join_context_properties(
format = "many_columns", vars = "all", include_description = TRUE))$traits

## End(Not run)
```

join_contributors *Joining data contributor metadata to traits table*

Description

Function to merge metadata from the data contributors table of a traits.build database into the core traits table.

Usage

```
join_contributors(database, format = "single_column_pretty", vars = "all")
```

Arguments

`database`	traits.build database (list object)
`format`	Specifies whether metadata from the contributors table is output in a human readable format ("single_column_pretty"; default) or using json syntax ("single_column_json").
`vars`	Columns from the taxa table to be joined to the traits table, defaulting to all columns (vars = "all").

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but with additional fields (columns) for the specified variables from the data contributors table appended to the traits table.

Examples

```
## Not run:
(database %>% join_contributors(format = "single_column_pretty",
vars = c("last_name", "first_name", "ORCID")))$traits

## End(Not run)
```

`join_location_coordinates` *Joining location coordinates to traits table*

Description

Function to merge geographic coordinates (latitude/longitude) stored in the locations table of a traits.build database into the core traits table.

Usage

```
join_location_coordinates(database)
```

Arguments

`database`	traits.build database (list object)

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but with additional fields (columns) for latitude and longitude appended to `traits` dataframe

Examples

```
## Not run:
(database %>% join_location_coordinates)$traits

## End(Not run)
```

`join_location_properties` *Joining location properties to traits table*

Description

Function to merge metadata from the locations table of a traits.build database into the core traits table.

Usage

```
join_location_properties(
  database,
  format = "single_column_pretty",
  vars = "all"
)
```

Arguments

`database`	traits.build database (list object)
`format`	Specifies whether metadata from the locations is output in a human readable format ("single_column_pretty"; default), with each location property added as a separate column ("many_columns") or using json syntax ("single_column_json").
`vars`	Location properties for which data is to be appended to the traits table, defaulting to all location properties (vars = "all").

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but location properties from the locations table appended to the traits table.

Examples

```
## Not run:
(database %>% join_location_properties(format = "single_column_pretty", vars = "all"))$traits

## End(Not run)
```

join_methods *Joining methodological information to traits table*

Description

Function to merge metadata from the methods table of a traits.build database into the core traits table.

Usage

```
join_methods(database, vars = c("methods"))
```

Arguments

`database`	traits.build database (list object)
`vars`	Columns from the taxa table to be joined to the traits table, defaulting to c("methods").

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but with additional fields (columns) for the specified variables from the methods table appended to the traits table.

Examples

```
## Not run:
(database %>% join_methods)$traits

## End(Not run)
```

join_taxa *Joining taxonomy to traits table*

Description

Function to merge metadata from the taxa table of a traits.build database into the core traits table.

Usage

```
join_taxa(
  database,
  vars = c("family", "genus", "taxon_rank", "establishment_means")
)
```

Arguments

database	traits.build database (list object)
vars	Columns from the taxa table to be joined to the traits table, defaulting to c("family", "genus", "taxon_rank", "establishment_means").

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: https://github.com/traitecoevo/traits.build & https://github.com/traitecoevo/traits.build-book

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see https://github.com/traitecoevo/austraits for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but with additional fields (columns) for the specified variables from the taxa table appended to the traits table.

Examples

```
## Not run:
#Append taxonomic details
(database %>% join_taxa)$traits

## End(Not run)
```

`join_taxonomic_updates` *Joining taxonomic updates information to traits table*

Description

Function to merge metadata from the taxonomic_updates table of a traits.build database into the core traits table.

Usage

```
join_taxonomic_updates(database, vars = c("aligned_name"))
```

Arguments

`database`	traits.build database (list object)
`vars`	Columns from the taxa table to be joined to the traits table, defaulting to c("aligned_name").

Details

the `join_` functions have been developed to join relational tables for databases built using the traits.build workflow. Learn more at: `https://github.com/traitecoevo/traits.build` & `https://github.com/traitecoevo/traits.build-book`

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see `https://github.com/traitecoevo/austraits` for how to install old versions of the package or download a newer version of the database.

Value

traits.build list object, but with additional fields (columns) for the specified variables from the taxonomic_updates table appended to the traits table.

Examples

```
## Not run:
#Append taxonomic update details
(database %>% join_taxonomic_updates)$traits

## End(Not run)
```

load_austraits — *Load AusTraits database into R console*

Description

Load AusTraits database into R console

Usage

```
load_austraits(
  doi = NULL,
  version = NULL,
  path = "data/austraits",
  update = FALSE
)
```

Arguments

doi	character string - doi of particular version
version	character string - version number of database
path	file path to where AusTraits will be downloaded
update	if TRUE, AusTraits versions .json will be re-downloaded

Value

a large list containing AusTraits data tables

See Also

get_versions get_version_latest

Examples

```
## Not run:
austraits <- load_austraits(version = "3.0.2", path = "data/austraits")

## End(Not run)
```

`lookup_context_property`
Look up context properties

Description

Look up context properties that contain a specific search term.

Usage

```
lookup_context_property(database, term)
```

Arguments

`database`	traits.build database (list object)
`term`	character string for context property search term

Value

vector containing context properties that contains search term

Examples

```
## Not run:
austraits %>% lookup_context_property("temperature")

## End(Not run)
```

`lookup_location_property`
Look up location properties

Description

Look up location properties that contain a specific search term.

Usage

```
lookup_location_property(database, term)
```

Arguments

`database`	traits.build database (list object)
`term`	character string for location property search term

Value

vector containing location properties that contains search term

Examples

```
## Not run:
austraits %>% lookup_location_property("soil")

## End(Not run)
```

`lookup_trait` *Look up a particular trait term*

Description

Look up a particular trait term

Usage

```
lookup_trait(database, term)
```

Arguments

`database`	traits.build database (list object)
`term`	character string for trait search term

Value

vector containing traits that contains search term

Examples

```
## Not run:
austraits %>% lookup_trait("leaf") %>% extract_trait(database = austraits, .)

## End(Not run)
```

`plot_locations` *Produce location maps of trait values*

Description

Plot location where trait data was collected from

Usage

```
plot_locations(database, feature = "trait_name", ...)
```

Arguments

`database`	traits.build database OR traits table from a traits.build database. Note location details must be joined. See join_location_coordinates and examples
`feature`	grouping/classification categories e.g trait_name, collection_type for <= v3.0.2, basis of record for >3.0.2
`...`	arguments passed to ggplot()

Value

ggplot of sites

Author(s)

Dony Indiarto - d.indiarto@student.unsw.edu.au

Examples

```
## Not run:
#All traits from a given study
data <- austraits %>% extract_dataset(dataset_id = "Falster_2003") %>% join_location_coordinates()
data %>% plot_locations("trait_name")

#Single trait
data <- austraits %>% extract_trait(trait_names = c("plant_height")) %>% join_location_coordinates()
data$traits %>% plot_locations("trait_name")

## End(Not run)
```

`plot_site_locations` *Produce location maps of trait values*

Description

[Deprecated]

Plot location where trait data was collected from

Usage

```
plot_site_locations(trait_data, feature = "trait_name", ...)
```

Arguments

`trait_data`	traits table in a traits.build database with site details appended. See join_location_coordinates and examples
`feature`	grouping/classification categories e.g trait_name, collection_type for <= v3.0.2
`...`	arguments passed to ggplot()

Value

ggplot of sites

Author(s)

Dony Indiarto - d.indiarto@student.unsw.edu.au

`plot_trait_distribution_beeswarm` *Beeswarm Trait distribution*

Description

Plots distribution of trait values by a grouping variable using ggbeeswarm package

Usage

```
plot_trait_distribution_beeswarm(
  database,
  trait_name,
  y_axis_category,
  highlight = NA,
  hide_ids = FALSE
)
```

Arguments

`database`	traits.build database (list object)
`trait_name`	Name of trait to plot
`y_axis_category`	One of `dataset_id`, `family`
`highlight`	Specify a group to highlight
`hide_ids`	Logical for whether to add a label on y_axis?

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Examples

```
## Not run:
austraits %>% plot_trait_distribution_beeswarm("wood_density", "dataset_id", "Westoby_2014")

## End(Not run)
```

`print.traits.build` *Generic for outputting a nice summary for austraits objects*

Description

Generic for outputting a nice summary for austraits objects

Usage

```
## S3 method for class 'traits.build'
print(x, ...)
```

Arguments

`x`	traits.build database
`...`	passed to print

Value

nicely printed table

`separate_trait_values` *Separate bounded trait values*

Description

This function reverts the action of bind_trait_values. This function separates values that were concatenated so that studies that have multiple observations for a given trait will have seperate row for each observation.

Usage

```
separate_trait_values(trait_data, definitions)
```

Arguments

`trait_data`	The traits table in a traits.build database - see example
`definitions`	The austraits definitions data frame

Value

trait tibble

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Examples

```
## Not run:
trait_data <- austraits$traits %>%
dplyr::filter(dataset_id == "Falster_2005_1")
trait_data
traits_bind <- bind_trait_values(trait_data)
separate_trait_values(traits_bind)

## End(Not run)
```

`summarise_database` *Summarise counts for a particular variable of interest*

Description

Summarise counts for a particular variable of interest

Usage

```
summarise_database(database, var)
```

Arguments

`database`	traits.build database (list object)
`var`	variable you use wish to see summary of (trait_name, genus, family)

Value

dataframe of unique levels of variable with counts and percentage

Examples

```
## Not run:
summarise_database(database = austraits, "trait_name")
summarise_database(database = austraits, "family")

## End(Not run)
```

`trait_pivot_longer` *Pivot wide format traits table into long format*

Description

[Deprecated] trait_pivot_longer "gathers" wide format data into a "tidy" format This function converts the data into long format where observations are on different rows and the type of observation is denoted by trait name. In other words, trait_pivot_longer reverts the actions of trait_pivot_wider

Usage

```
trait_pivot_longer(wide_data)
```

Arguments

`wide_data`	output from trait_pivot_wider.

Details

- If `bind_trait_values` was applied prior to `trait_pivot_wider` for AusTraits <= v3.0.2, `trait_pivot_longer` will return a tibble with fewer observations than the original traits table.
- For AusTraits version >3.0.2, `trait_pivot_longer` will return a tibble with fewer columns than that original traits table
  - The excluded columns include: "unit", "replicates", "measurement_remarks", "basis_of_record", "basis_of_value"

This function reverts the actions of the function austraits::trait_pivot_wider.

It begins with a derivation of a traits.build traits table, where multiple measurements that comprise a single observation are displayed on a single row,with a column for each trait. It then converts the table into long format where measurements of multiple traits that comprise a single observation are on different rows and a column specifying the trait names is added.

`trait_pivot_longer` has been developed to pivot the traits table for a database build using the traits.build workflow. Learn more at: https://github.com/traitecoevo/traits.build & https://github.com/traitecoevo/traits.build-book

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see https://github.com/traitecoevo/austraits for how to install old versions of the package or download a newer version of the database.

Value

A tibble in long format

A tibble in long format

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Fonti Kar - fonti.kar@unsw.edu.au

Examples

```
## Not run:
data <- austraits$traits %>%
filter(dataset_id == "Falster_2003")
data #long format
traits_wide <- trait_pivot_wider(data)
traits_wide #wide format

values_long <- trait_pivot_longer(traits_wide)

## End(Not run)
```

`trait_pivot_wider` *Pivot long format traits table into wide format*

Description

Function to "widen" long format data ("tidy data").

Data in a traits.build databases' traits table are organised in a long format where each trait measurement is on a different row and measurement metadata is recorded in other columns. Multiple traits may be measured as part of a single observation and this function pivots the data wider, such that each trait is its own column. Note that if two trait measurements have the same observation_id but different value types (min, mean, mode, etc.) these will be on separate rows.

The function austraits::trait_pivot_longer reverts the actions of this function.

Usage

```
trait_pivot_wider(database)
```

Arguments

`database`	The traits tibble from a traits.build database

Details

'trait_pivot_wider" has been developed to pivot the traits table for a database build using the traits.build workflow. Learn more at: https://github.com/traitecoevo/traits.build & https://github.com/traitecoevo/traits.build-book

Note to AusTraits users:

- This function works with AusTraits version >= 5.0.0 (from Nov 2023 release)
- For AusTraits versions <= 4.2.0 (up to Sept 2023 release) see https://github.com/traitecoevo/austraits for how to install old versions of the package or download a newer version of the database.

Value

traits.build traits table in wide format

Author(s)

Daniel Falster - daniel.falster@unsw.edu.au

Examples

```
## Not run:

data <- austraits_5.0.0_lite$traits %>% filter(dataset_id == "Falster_2003")
data #long format
traits_wide <- trait_pivot_wider(data)
traits_wide #wide format

## End(Not run)
```

Index