Package ‘traitstrap’

May 12, 2023

Title Bootstrap Trait Values to Calculate Moments

Version 0.1.0

Description Calculates trait moments from trait and community data using the methods developed in Maitner et al (2021) <doi:10.22541/au.162196147.76797968/v1>.

URL `https://github.com/Plant-Functional-Trait-Course/traitstrap/`

BugReports `https://github.com/Plant-Functional-Trait-Course/traitstrap/issues`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 4.2.0)

Imports dplyr (>= 1.0.0), purrr, rlang, e1071, glue, ggplot2, tibble, stringr, fitdistrplus, tidyr

Suggests testthat, knitr, rmarkdown, FD, TPD

VignetteBuilder knitr

NeedsCompilation no

Author Richard J. Telford [aut, cre] (<`https://orcid.org/0000-0001-9826-3076`>), Aud H. Halbritter [aut] (<`https://orcid.org/0000-0003-2597-6328`>), Brian S. Maitner [aut] (<`https://orcid.org/0000-0002-2118-9880`>)

Maintainer Richard J. Telford <`richard.telford@uib.no`>

Repository CRAN

Date/Publication 2023-05-12 14:10:02 UTC

R topics documented:

`autoplot.filled_trait` *Coverage plot of filled Traits*

Description

Function calculates the trait coverage of the community for each level of the sampling hierarchy and makes a barplot.

Shows at which level the data are coming from in each plot.

Usage

```
## S3 method for class 'filled_trait'
autoplot(object, other_col_how, ...)
```

Arguments

`object`	output from `trait_fill()`.
`other_col_how`	what to do with the other columns in other data. Options are to `filter` by one of the columns, add them to the `x-axis`, `facet` by them, or to `ignore`.
`...`	optional filters for use with `other_col_how` = `"filter"`

Value

a ggplot2 plot

Examples

```
require("ggplot2")
data(community)
data(trait)
filled_traits <- trait_fill(
  comm = community, traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)
autoplot(filled_traits)
```

`community` *Community data*

Description

A dataset containing plant cover in control plots on Svalbard from PFCT4 TraitTrain course.

Usage

```
community
```

Format

A data frame with 110 rows and 4 variables:

Taxon species name

Cover cover, in percent

Site site name

PlotID plot name

Source

`https://www.uib.no/en/rg/EECRG/114808/plant-functional-traits-course-4`

`cor_to_df` *Correlation to dataframe*

Description

Helper function for bootstrap_traits_multivariate that extracts results from a correlation matrix

Usage

```
cor_to_df(corr)
```

Arguments

`corr`	correlation matrix

Value

A data.frame of correlations

References

Modified from https://stackoverflow.com/a/23476844/2055765

Examples

```
x <- matrix(ncol = 5, rnorm(20))
colnames(x) <- letters[1:5]
cor(x) |> cor_to_df()
```

`fortify_filled_trait` *Fortify filled Traits*

Description

Calculates at which level the data are coming from in each plot.

Usage

```
fortify_filled_trait(object, other_col_how, ...)
```

Arguments

`object`	filled traits from `trait_fill()`.
`other_col_how`	what to do with the other columns in other data. Options are to `filter` by one of the columns, add them to the `x-axis`, `facet` by them, or to `ignore`.
`...`	optional filters for use with `other_col_how = "filter"`

Value

a tibble

`trait` *Trait data*

Description

A dataset containing plant traits in control plots on Svalbard from PFCT4 TraitTrain course.

Usage

```
trait
```

Format

A data frame with 705 rows and 6 variables:

Site site name

PlotID plot name

Taxon species name

ID Unique leaf ID

Trait trait name with unit

Value trait value

Source

https://www.uib.no/en/rg/EECRG/114808/plant-functional-traits-course-4

`trait_fill` *fill traits*

Description

A function for trait filling using a hierarchical sampling design, which allows to account for incomplete trait collections, traits from different spatial or temporal levels (i.e. local traits vs. databases) and experimental designs.

Usage

```
trait_fill(
  comm,
  traits,
  scale_hierarchy = c("Country", "Site", "BlockID", "PlotID"),
  global = TRUE,
  taxon_col = "taxon",
  trait_col = "trait",
  value_col = "Value",
  abundance_col = "Cover",
  treatment_col = NULL,
  treatment_level = NULL,
  other_col = character(0),
  keep_all = FALSE,
  min_n_in_sample = 5,
  complete_only = FALSE,
  leaf_id
)
```

Arguments

`comm`	a dataframe in long format with community data
`traits`	a dataframe in long format with trait data
`scale_hierarchy`	character vector of the sampling hierarchy from large to small (e.g. site/block/plot)
`global`	logical; calculate traits at global scale. Must not be a column called global in the traits data.
`taxon_col`	character; name of taxon column in comm and traits. Can be a vector (e.g. "species", "genus"), in which case if traits cannot be filled for the first taxon column, subsequent columns will be used in order.
`trait_col`	character; name of trait name column in traits
`value_col`	character; name of trait value column in traits
`abundance_col`	character; name of species abundance column in comm
`treatment_col`	character; optional name of treatment_col in comm and traits. Must refer to a factor where first level is control.
`treatment_level`	character; optional name of scale_hierarchy at which treatment should be filtered
`other_col`	name of other grouping columns in comm
`keep_all`	logical; keep trait data at all available levels or just finest scale available
`min_n_in_sample`	numeric; minimum number in sample with traits to accept before searching for traits higher up the hierarchy. The default is 5.
`complete_only`	logical; use only leaves with a full set of traits. Set to TRUE when imputing for `trait_multivariate_bootstrap`
`leaf_id`	character; unique leaf identifiers. Only needed when `complete_only` is TRUE.

Details

The function uses a hierarchical sampling design, which allows it to account for incomplete trait collections, traits from different spatial or temporal levels (i.e. local traits vs. databases) and/or experimental designs.

With scale_hierarchy you can define the levels at which the traits have been collected and their order starting with the highest level (e.g. global database, region, site, block, plot).

`trait_fill()` will choose if available a trait value from the lowest level, i.e. species X from plot A and if no trait is available from that level, it will move up the hierarchy and choose a trait from species X from plot B at the same site. If there is no trait available from species X in the same site, it will choose a trait value from another site.

The argument `min_n_in_samples` allows users to define the minimum number in sample at each level for the trait filling. If the minimum number is not reached, trait values from the next level will also be selected, to avoid sampling the same individual several times, which will result in unrealistic variances. The default value is 5.

In the `other_col` argument other grouping variables in the community dataset can be defined and will be kept after the trait filling step.

Traitstrap also allows to include taxonomy and experimental design in the trait filling step.

With taxon_col a hierarchy for the taxonomy can be defined. If traits for a specific species are not available, traits from next level, e.g. the genus will be selected. For this a list of the taxonomic hierarchy has to be defined (e.g. "taxon", "genus", "family").

The argument treatment_col allows to incorporate an experimental design where traits are selected from the same experimental treatment or the first factor level, which is assumed to be the control. Therefore, it is important to order the levels of a treatment in the right order, i.e. the first level has to be the control. If you have two or more treatments and you want filling to be done only within a treatment, and not from a treatment and the control, then make the first level of the factor a level that is not in the data. The filling step can be defined at certain level using the treatment_level argument. Depending on the experimental design trait filling should occur a certain level, e.g. block or site.

Value

a tibble with extra class `filled_trait`.

Examples

```
data(community)
data(trait)
filled_traits <- trait_fill(
  comm = community, traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)
```

`trait_fit_distributions` *Fit trait distributions*

Description

Function to fit parametric distributions for each species-by-trait combination at the finest scale of the user-supplied hierarchy. This function returns a tibble containing the fitted parameters.

Usage

```
trait_fit_distributions(filled_traits, distribution_type = "normal")
```

Arguments

`filled_traits`	output from the trait_fill function.
`distribution_type`	the type of statistical distribution to use. Character or named list. Currently accepts "normal","lognormal", and "beta".

Details

The distributions can either be a single distribution type which is used for all traits, or traits can be assigned specific distributions types by supplying the function with a named vector of traits, e.g. `c(height = "normal", mass = "lognormal"))`.

Value

a tibble containing fitted distribution parameters for each trait in each species for each plot.

Examples

```
library(dplyr)
data(community)
data(trait)

filled_traits <- trait_fill(
  comm = community |>
    filter(PlotID %in% c("A", "B")),
  traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)

fitted_distributions <- trait_fit_distributions(
  filled_traits = filled_traits,
  distribution_type = "normal"
)
```

`trait_missing` *Which taxa lack traits*

Description

Function gives overview of which taxa are missing traits.

Usage

```
trait_missing(filled_trait, comm)
```

Arguments

`filled_trait`	output of trait_fill function.
`comm`	community data

Value

A `tibble` with columns #'

- Taxon Species names (actual name depends on `taxon_col` argument to `trait_fill()`)
- max_abun Maximum abundance of that taxa. Be more concerned about taxa missing traits with high abundances.
- n Number of occurrences of the taxon. Be more concerned about taxa missing traits with many occurrences.
- n_traits Number of traits for each species. Ideally all should equal the number of traits you have measured.

Examples

```
data(community)
data(trait)
filled_traits <- trait_fill(
  comm = community, traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)
trait_missing(filled_traits, community)
```

`trait_multivariate_bootstrap`

Bootstrap traits

Description

Function for nonparametric bootstrap resampling to calculate community weighted trait correlations, other bivariate or multivariate statistics

Usage

```
trait_multivariate_bootstrap(
  filled_traits,
  nrep = 100,
  sample_size = 200,
  raw = FALSE,
  id = "ID",
  fun
)
```

Arguments

`filled_traits`	output from the trait_fill function.
`nrep`	number of bootstrap replicates
`sample_size`	bootstrap size
`raw`	logical; argument to extract the raw data of the trait distributions. The default is `raw = FALSE`. If `raw = TRUE`, `nrep` is restricted to 1 to avoid memory issues.
`id`	column name of unique identifiers of each leaf
`fun`	bivariate or multivariate function to apply

Details

The observed and filled leaves are re-sampled in proportion to their weights, e.g. the abundance of a species or the biomass. Values across all individuals in a community are resampled `sample_size` times to incorporate the full spectrum of trait variation, generating `nrep` trait distributions. The function `fun` is applied to the trait distribution at the finest level of the filled trait hierarchy.

Note that due to the flexibility of this function, the output CAN NOT be summarized using `trait_summarise_boot_moments`

Value

a tibble with columns for the elements of the scale_hierarchy, and a list column result which includes the output of `fun`.

Examples

```
require(dplyr)
require(tidyr)
require(ggplot2)
require(purrr)

data(community)
data(trait)

filled_traits <- trait_fill(
  comm = community |>
    filter(
      PlotID %in% c("A", "B"),
      Site == 1
    ),
  traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover",
  complete_only = TRUE, leaf_id = "ID"
)

# Note that more replicates and a greater sample size are advisable
# Here we set them low to make the example run quickly
boot_traits <- trait_multivariate_bootstrap(filled_traits,
  fun = cor,
```

```
  nrep = 10,
  sample_size = 100
)

boot_traits_long <- boot_traits |>
  mutate(correlations = map(result, ~ cor_to_df(.x))) |>
  select(-result) |>
  unnest(correlations)

boot_traits_long |>
  ggplot(aes(x = paste(row, "v", col), y = value)) +
  geom_violin() +
  facet_grid(Site ~ PlotID) +
  coord_flip() +
  labs(y = "Correlation", x = "")
```

`trait_np_bootstrap` *Bootstrap traits*

Description

Function for nonparametric bootstrap resampling to calculate community weighted trait mean and higher moments.

Usage

```
trait_np_bootstrap(filled_traits, nrep = 100, sample_size = 200, raw = FALSE)
```

Arguments

`filled_traits`	output from the trait_fill function.
`nrep`	number of bootstrap replicates
`sample_size`	bootstrap size
`raw`	logical; argument to extract the raw data of the trait distributions. The default is `raw = FALSE`. If `raw = TRUE`, `nrep` is restricted to 1 to avoid memory issues.

Details

The observed traits are re-sampled in proportion to their weights, e.g. the abundance of a species or the biomass. Values across all individuals in a community are resampled `sample_size` times to incorporate the full spectrum of trait variation, generating `nrep` trait distributions. From these distributions the function estimates the mean and the higher moments including variance, skewness and kurtosis.

#' The output of `trait_np_bootstrap()` can be summarized using `trait_summarize_boot_moments()`.

Value

a tibble with columns for each grouping variable of `filled_traits` (usually the elements of scale_hierarchy and the traits column), and the moments mean, variance, skewness, and kurtosis.

Examples

```
library(dplyr)
data(community)
data(trait)

# Filter community data to make example faster
community <- community |>
  filter(
    PlotID %in% c("A", "B"),
    Site == 1
  )
filled_traits <- trait_fill(
  comm = community,
  traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)

boot_traits <- trait_np_bootstrap(filled_traits,
  nrep = 20,
  sample_size = 200
)
```

`trait_parametric_bootstrap` *Bootstrap traits parametrically*

Description

Function for parametric bootstrap resampling to calculate community weighted trait mean and higher moments.

Usage

```
trait_parametric_bootstrap(
  fitted_distributions,
  nrep = 100,
  sample_size = 200,
  raw = FALSE
)
```

Arguments

`fitted_distributions`	Fitted distribution object returned by trait_fit_distributions
`nrep`	number of bootstrap replicates
`sample_size`	bootstrap size

`raw`	logical; argument to extract the raw data of the trait distributions. The default is `raw = FALSE`. If `raw = TRUE`, `nrep` is restricted to 1 to avoid memory issues.

Details

`trait_parametric_bootstrap()` is a parametric analogue of the `trait_np_bootstrap()`. It randomly samples from among the fitted distributions proportionally to species abundance. The number of samples per replicated are drawn specified with the parameter sample_size, and the number of replicates is specified by the parameter `nrep`. From these distributions the function estimates the mean and the higher moments including variance, skewness and kurtosis.

The output of `trait_parametric_bootstrap()` can be summarized using `trait_summarize_boot_moments()`.

Value

a tibble

Examples

```
library(dplyr)
data(community)
data(trait)

# Filter trait and community data to make example faster

community <- community |>
  filter(
    PlotID %in% c("A", "B"),
    Site == 1
  )

trait <- trait |>
  filter(Trait %in% c("Plant_Height_cm"))

filled_traits <- trait_fill(
  comm = community,
  traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)

fitted_distributions <- trait_fit_distributions(
  filled_traits = filled_traits,
  distribution_type = "normal"
)

# Note that more replicates and a greater sample size are advisable
# Here we set them low to make the example run quickly
parametric_distributions <- trait_parametric_bootstrap(
  fitted_distributions = fitted_distributions,
  nrep = 5,
  sample_size = 100
```

```
)

moment_summary <- trait_summarise_boot_moments(
  bootstrap_moments = parametric_distributions,
  parametric = FALSE
)
```

`trait_summarise_boot_moments`

Summarise Bootstrap traits

Description

Summarizes the mean and confidence interval for each trait moment.

Usage

```
trait_summarise_boot_moments(
  bootstrap_moments,
  parametric = TRUE,
  sd_mult = 1,
  ci = 0.95
)
```

Arguments

`bootstrap_moments`	trait moments from trait_np_bootstrap or trait_parametric_bootstrap
`parametric`	logical; default is TRUE. Should Confidence Intervals be calculated parametrically (using the mean and SD) or nonparametrically (using quantiles).
`sd_mult`	Number of standard deviations around each moment, defaults to one
`ci`	Desired confidence level for use when parametric is false. Defaults to 0.95.

Value

tibble with the grouping variables and the mean of each moment (+/- sd_mult * SD)

Examples

```
library(dplyr)
data(community)
data(trait)

# Filter community data to make example faster
community <- community |>
  filter(PlotID %in% c("A", "B"))

filled_traits <- trait_fill(
```

```
  comm = community,
  traits = trait,
  scale_hierarchy = c("Site", "PlotID"),
  taxon_col = "Taxon", value_col = "Value",
  trait_col = "Trait", abundance_col = "Cover"
)

# Note that more replicates and a greater sample size are advisable
# Here we set them low to make the example run quickly
boot_traits <- trait_np_bootstrap(filled_traits,
  nrep = 20,
  sample_size = 100
)

trait_summarise_boot_moments(boot_traits)
```

Index