Package: nichetools (via r-universe)

October 10, 2026

Type Package

Title Complementary Package to 'nicheROVER' and 'SIBER'

Version 0.3.3

Description Provides functions complementary to packages 'nicheROVER' and 'SIBER' allowing the user to extract Bayesian estimates from data objects created by the packages 'nicheROVER' and 'SIBER'. Please see the following publications for detailed methods on 'nicheROVER' and 'SIBER' Hansen et al. (2015) <doi:10.1890/14-0235.1>, Jackson et al. (2011) <doi:10.1111/j.1365-2656.2011.01806.x>, and Layman et al. (2007) <doi:10.1890/0012-9658(2007)88[42:CSIRPF]2.0.CO;2>, respectfully.

Depends R (>= 4.1.0)

Imports cli, dplyr, ellipse, lifecycle, nicheROVER, purrr, rlang, SIBER, tibble, tidyr

License CC0

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://benjaminhlina.github.io/nichetools/, https://github.com/benjaminhlina/nichetools

BugReports https://github.com/benjaminhlina/nichetools/issues

Suggests bayestestR, ggplot2, ggdist, ggtext, ggh4x, janitor, knitr, patchwork, rjags, rmarkdown, stringr, testthat (>= 3.0.0), viridis

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs jags libicu-dev

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

Date/Publication 2026-01-13 18:09:20 UTC

RemoteUrl https://github.com/benjaminhlina/nichetools

RemoteRef HEAD

RemoteSha ad51e49f646dc9ddce88856ee10d42af2a83b25d

Contents

create_comparisons 2
extract_group_metrics 3
extract_layman 4
extract_mu 7
extract_niche_size 8
extract_overlap 10
extract_sigma 10
extract_similarities 12
mu_est_long 13
niche_ellipse 14
niw_fish_post 15
over_stat 16
post_sam_siber 16
sigma_est_wide 17

Index **18**

`create_comparisons` *create comparisons*

Description

Creates a list with all of the comparisons needed to create Bayesian and maximum-likelihood estimates for proportion of niche similarities.

Usage

```
create_comparisons(data, comparison = c("within", "among"))
```

Arguments

`data`	a `data.frame` that is the names of the community and group names
`comparison`	a `character`that is either `"within"` or `"among"` indicating whether the comparisons are within a community and between groups or among communities for the same groups.

Examples

```
# ---- load siber ----
library(SIBER)

# ---- create community names data frame ----
# uncomment to use
# str(demo.siber.data.2)

demo.siber.data.2$group_name <- as.factor(demo.siber.data.2$group)

demo.siber.data.2$group <- as.numeric(demo.siber.data.2$group_name) |>
as.character()

demo.siber.data.2$community_names <- as.factor(demo.siber.data.2$community)

demo.siber.data.2$community <- as.numeric(demo.siber.data.2$community_names) |>
as.character()

cg_names <- demo.siber.data.2 |>
dplyr::distinct(community, group, community_names, group_name)

# ---- create comparsions ----
create_comparisons(cg_names,
                   comparison = "within")
```

`extract_group_metrics` *extract maximum-likelihood estimates for group metrics*

Description

Extract group metrics within each community from a matrix object that is produced by `groupMetricsML()` function from SIBER. These metrics are the following the convex hull total area (TA), Standard Ellipse Area (SEA), and the corresponding small sample size corrected version SEAc based on the maximum likelihood estimates of the means and covariance matrices of each group.

Usage

```
extract_group_metrics(data = NULL, community_df = NULL, data_format = NULL)
```

Arguments

`data`	a `matrix` produced by the function `groupMetricsML()` in the package SIBER.
`community_df`	a four column data frame. One of the columns has to be named `community` and the data in the column will be `numeric` as a `character` string(e.g., `"1"`, `"2"`, `"3"`). This is the order of the community names and will be used to join the actual community names to the correct data. These are the same class and values required by the function, `createSiberObject()` from SIBER. The second column will

be the names of the groups that are needed to supply required by the function, `createSiberObject()` from SIBER. The third and fourth columns contains the actual names of the communities and groups the user is working with (e.g., `"region"`, `"common_name"`).

`data_format` a `character` string that decides whether the returned object is in long or wide format. Default is `"long"`, with the alternative supplied being `"wide"`.

Value

A `tibble` containing four rows when `data_format` is set to its default which is `long`. These four rows are the following, `community`, `the_name_of_the_communities`, `metric` and `post_est`.

Examples

```
library(SIBER)

# ---- create community names data frame ----
# uncomment to use
# str(demo.siber.data.2)

demo.siber.data.2$group_name <- as.factor(demo.siber.data.2$group)

demo.siber.data.2$group <- as.numeric(demo.siber.data.2$group_name) |>
as.character()

demo.siber.data.2$community_name <- as.factor(demo.siber.data.2$community)

demo.siber.data.2$community <- as.numeric(demo.siber.data.2$community_name) |>
as.character()

cg_name <- demo.siber.data.2 |>
dplyr::distinct(community, group, community_name, group_name)

# ---- create comparsions ----

demo.siber.data.2 <- demo.siber.data.2[,1:4]

siber_example <- createSiberObject(demo.siber.data.2)

# extract group metrics
group_ml <- groupMetricsML(siber_example)

group_convert <- extract_group_metrics(data = group_ml,
                                community_df = cg_name)
```

`extract_layman` *extract Layman metrics*

Description

Extract Bayesian estimates for the following six layman metrics, δ^{13}C range, δ^{15}N range, total area (TA), distance to centroid (CD), distance to the nearest neighbour (NND), and the standard deviation of the distance to the nearest neighbour (SDNND) from data objects created by SIBER. To learn more about the following metrics please review Layman et al. (2008).

Usage

```
extract_layman(
  data,
  type = NULL,
  community_df = NULL,
  data_format = NULL,
  isotope_x = NULL,
  isotope_y = NULL,
  element_x = NULL,
  element_y = NULL
)
```

Arguments

`data`	a `list` created by the function `bayesianLayman()` from the package SIBER.
`type`	a `character` that is either `"bay"` or `"ml"` which indicates whether the community metrics to be extracted are from a Bayesian analysis or a maximum-likelihood.
`community_df`	a two column data frame. One of the columns has to be named `community` and the data in the column will be `numerics` as a `character` string(e.g., `"1"`, `"2"`, `"3"`). This is the order of the community names and will be used to join the actual community names to the correct data. These are the same class and values required by the function, `createSiberObject()` from SIBER. The second column contains the actual names of the communities that the user is working with (e.g., `"region"`).
`data_format`	a `character` string that decides whether the returned object is in long or wide format. Default is `"long"`, with the alternative supplied being `"wide"`.
`isotope_x`	a `numeric` that will be used in the labeling processes for the range of the x. Default is `13` (e.g., δ^13 C). This will create a column called `labels`, that will only be created when `data_format` is set to `long`.
`isotope_y`	a `numeric` that will be used in the labeling processes for the range of the y isotope. Default is `15` (e.g., δ^15 N). #' This will create a column called `labels`, that will only be created when `data_format` is set to `long`.
`element_x`	a `character` that will be used in the labeling process for the range of the x isotope. Default is `C` (e.g., δ^13 C). This will create a column called `labels`, that will only be created when `data_format` is set to `long`.
`element_y`	a `character` that will be used in the labeling process for the range of the y isotope. Default is `N` (e.g., δ^13 N). #' This will create a column called `labels`, that will only be created when `data_format` is set to `long`.

Value

A `tibble` containing four rows when `data_format` is set to its default which is `long`. These four rows are the following, `community`, `the_name_of_the_communities`, `metric` and `post_est`.

See Also

`SIBER::bayesianLayman()` and `SIBER::createSiberObject()`

Examples

```
library(SIBER)

# ---- bring in SIBER demo data ----
# uncomenet to use
# str(demo.siber.data)

# ---- create community names data frame ----
# uncomment to use
# str(demo.siber.data.2)

demo.siber.data.2$group_name <- as.factor(demo.siber.data.2$group)

demo.siber.data.2$group <- as.numeric(demo.siber.data.2$group_name) |>
as.character()

demo.siber.data.2$community_names <- as.factor(demo.siber.data.2$community)

demo.siber.data.2$community <- as.numeric(demo.siber.data.2$community_names) |>
as.character()
c_names <- demo.siber.data.2 |>
dplyr::distinct(community, community_names)


demo.siber.data_2 <- demo.siber.data.2[,1:4]
# ---- create the siber object ----
siber.example <- createSiberObject(demo.siber.data_2)

# ---- view Bayesian estimates of mu and sigma produced by SIBER ---
# uncomment to use
# str(post_sam_siber)

# ---- extract posterior estimates of mu -----

mu_post <- extractPosteriorMeans(siber.example, post_sam_siber)

# ---- Bayesian estimates of layman metrics using SIBER ----

layman_b <- bayesianLayman(mu.post = mu_post)

# ---- use nichetools to extract Bayesian estimates of Layman metrics ----

layman_be <- extract_layman(layman_b, community_df = c_names)
```

layman_be

---

extract_mu *extract μ*

Description

Extract Bayesian estimates of μ from data objects created by nicheROVER or SIBER.

Usage

```
extract_mu(
  data,
  pkg = NULL,
  isotope_names = NULL,
  data_format = NULL,
  community_df = NULL
)
```

Arguments

data	a list created by the function niw.post() or siberMVN() in the package nicheROVER or SIBER, respectfully.
pkg	a character string that is the name of the package that you're using. Defaults to "nicheROVER". Alternatively the user can supply the argument with "SIBER".
isotope_names	is a vector of character string used change the column name of isotopes used in the analysis. Defaults to c("d13c", "d15n").
data_format	a character string that decides whether the returned object is in long or wide format. Default is "long", with the alternative supplied being "wide".
community_df	a four column data frame. One of the columns has to be named community and the data in the column will be numeric as a character string(e.g., "1", "2", "3"). This is the order of the community names and will be used to join the actual community names to the correct data. These are the same class and values required by the function, createSiberObject() from SIBER. The second column will be the names of the groups that are needed to supply required by the function, createSiberObject() from SIBER. The third and fourth columns contains the actual names of the communities and groups the user is working with (e.g., "region", "common_name").

Value

Returns a tibble of extracted estimates of μ created by the function niw.post() or siberMVN() in the packages nicheROVER. and SIBER.

The tibble will contain five columns in the following order, metric, sample_name, sample_number, and the names of the isotope columns supplied to niw.post() or siberMVN() (e.g., d13c and d15n).

See Also

nicheROVER::niw.post() and SIBER::siberMVN()

Examples

```
extract_mu(
data = niw_fish_post
)

library(SIBER)

# ---- create community names data frame ----
# uncomment to use
# str(demo.siber.data.2)

demo.siber.data.2$group_name <- as.factor(demo.siber.data.2$group)

demo.siber.data.2$group <- as.numeric(demo.siber.data.2$group_name) |>
as.character()

demo.siber.data.2$community_name <- as.factor(demo.siber.data.2$community)

demo.siber.data.2$community <- as.numeric(demo.siber.data.2$community_name) |>
as.character()

cg_name <- demo.siber.data.2 |>
dplyr::distinct(community, group, community_name, group_name)


extract_mu(
data = post_sam_siber,
pkg = "SIBER",
community_df = cg_name
)
```

---

extract_niche_size *extract niche size*

Description

Extract niche size based on elliptical niche region of Bayesian estimates of sigma created by function `niw.post()` or `siberEllipses()` in the package nicheROVER or SIBER, respectfully. For nicheROVER this function is a wrapper around `nicheROVER::niche.size`.

Usage

```
extract_niche_size(
  data,
```

```
  pkg = NULL,
  name = NULL,
  prob = NULL,
  community_df = NULL
)
```

Arguments

data	a list or matrix created by the function niw.post() or siberEllipses() in the package nicheROVER or SIBER, respectfully.
pkg	a character string that is the name of the package that you're using. Defaults to "nicheROVER". Alternatively the user can supply the argument with "SIBER".
name	a character string that will be assigned as the column name for groups. Default is sample_name. Only to be used when pkg is set to "nicheROVER".
prob	a numeric bound by 0 and 1 indicating the probabilistic niche size. Default is 0.95. Only to be used when pkg is set to "nicheROVER".
community_df	a four column data frame. One of the columns has to be named community and the data in the column will be numeric as a character string(e.g., "1", "2", "3"). This is the order of the community names and will be used to join the actual community names to the correct data. These are the same class and values required by the function, createSiberObject() from SIBER. The second column will be the names of the groups that are needed to supply required by the function, createSiberObject() from SIBER. The third and fourth columns contains the actual names of the communities and groups the user is working with (e.g., "region", "common_name").

Value

if pkg is set to "nicheROVER" then a tibble containing three rows, sample_name, id, and niche_size is returned.

See Also

nicheROVER::niche.size(), nicheROVER::niw.post(), SIBER::siberEllipses(), and SIBER::createSiberObject(

Examples

```
extract_niche_size(data = niw_fish_post)
```

extract_overlap *extract overlap*

Description

Extract Bayesian estimates of similarities among groups produced by the following function `overlap()` in the package nicheROVER.

Usage

```
extract_overlap(data, name_a = NULL, name_b = NULL)
```

Arguments

`data`	a `array` object containing `matrices` created by the function `overlap()` in the package nicheROVER.
`name_a`	character string to supply for the first `sample_name` used in `overlap()`. Defaults to `"sample_name_a"`.
`name_b`	character string to supply for the second `sample_name` used in `overlap()`. Defaults to `"sample_name_b"`.

Value

A `tibble` containing five rows, `sample_name_a`, `id`, `sample_name_b`, `sample_number`, and `niche_overlap`.

See Also

`nicheROVER::overlap()`

Examples

```
extract_overlap(data = over_stat)
```

extract_sigma *extract* Σ

Description

Extract Bayesian estimates of Σ from data objects created by nicheROVER or SIBER.

Usage

```
extract_sigma(
  data,
  pkg = NULL,
  isotope_n = NULL,
  isotope_names = NULL,
  data_format = NULL
)
```

Arguments

data	a list created by the function niw.post() or siberMVN() in the package nicheROVER or SIBER, respectfully.
pkg	a character string that is the name of the package that you're using. Defaults to "nicheROVER". Alternatively the user can supply the argument with "SIBER".
isotope_n	a numeric either 2 or 3 that is the number of isotopes used in the anlsysis. Will default to 2.
isotope_names	is a vector of character string used change the column name of isotopes used in the analysis. Defaults to c("d13c", "d15n").
data_format	a character string that decides whether the returned object is in long or wide format. Default is "wide", with the alternative supplied being "long".

Value

Returns a tibble of extracted estimates of Σ created by the function niw.post() or siberMVN() in the packages nicheROVER. and SIBER.

The returned object will contain five columns in the following order when data_format is set to "wide", metric, id, sample_name, isotope, sample_number, and the posterior sample for Σ (e.g., d13c and d15n).

See Also

nicheROVER::niw.post() and SIBER::siberMVN()

Examples

```
extract_sigma(
data = niw_fish_post
)

extract_sigma(
data = post_sam_siber,
pkg = "SIBER"
)
```

`extract_similarities` *extract similarities*

Description

Extract niche similarities from objects created by {SIBER}.

Usage

```
extract_similarities(data, type = c("bay", "ml"), community_df = NULL)
```

Arguments

data	a list of results from either maxLikOverlap() or bayesianOverlap().
type	a character that is either "bay" or "ml" which indicates whether the community metrics to be extracted are from a Bayesian analysis or a maximum-likelihood.
community_df	a four column data frame. One of the columns has to be named community and the data in the column will be numeric as a character string(e.g., "1", "2", "3"). This is the order of the community names and will be used to join the actual community names to the correct data. These are the same class and values required by the function, createSiberObject() from SIBER. The second column will be the names of the groups that are needed to supply required by the function, createSiberObject() from SIBER. The third and fourth columns contains the actual names of the communities and groups the user is working with (e.g., "region", "common_name").

Examples

```
library(purrr)
library(SIBER)

# ---- create community names data frame ----
# uncomment to use
# str(demo.siber.data.2)

demo.siber.data.2$group_name <- as.factor(demo.siber.data.2$group)

demo.siber.data.2$group <- as.numeric(demo.siber.data.2$group_name) |>
as.character()

demo.siber.data.2$community_name <- as.factor(demo.siber.data.2$community)

demo.siber.data.2$community <- as.numeric(demo.siber.data.2$community_name) |>
as.character()

cg_name <- demo.siber.data.2 |>
dplyr::distinct(community, group, community_name, group_name)
```

```
# ---- create comparsions ----
cg_names_within_c <- create_comparisons(cg_name,
                                        comparison = "within")

demo.siber.data.2 <- demo.siber.data.2[,1:4]

siber_example <- createSiberObject(demo.siber.data.2)

ml_within_overlap <- cg_names_within_c |>
map(~ maxLikOverlap(.x$cg_1, .x$cg_2, siber_example,
p.interval = NULL, n = 100), .progress = TRUE)

ml_95_within_com <- extract_similarities(ml_within_overlap, type = "ml",
community_df = cg_name)
```

`mu_est_long` *A* `data.frame` *containing posterior estimates of* μ

Description

Posterior estimates of μ using `fish` data set from nicheROVER, using Normal-Inverse-Wishart (NIW) priors.

Usage

```
mu_est_long
```

Format

`data.frame` containing 8,000 rows and 7 variables

metric name of the metric extracted from `niw.post()`

species species abbreviation

sample_number sample number from 1-1000

isotope column with isotope name

mu_est estimate of mu produced from `niw.post()`

element isotopic element used in labelling

neutron neutron number used in labelling

niche_ellipse *Create ellipses based on Bayesian estimates of μ and Σ*

Description

This function allows the user to supply Bayesian estimates of μ and Σ to create estimated Bayesian ellipse for niche region.

Usage

```
niche_ellipse(
  dat_mu,
  dat_sigma,
  isotope_a = NULL,
  isotope_b = NULL,
  p_ell = NULL,
  random = NULL,
  set_seed = NULL,
  n = NULL,
  message = TRUE,
  ...
)
```

Arguments

dat_mu	a data.frame containing μ Bayesian estimates. This data.frame needs to be in long format with each μ estimate for each isotope stacked on top of each other. This can be produced using extract_mu().
dat_sigma	a data.frame containing Σ Bayesian estimates. This data.frame needs be in wide format, that is Σ (covariance) matrices stacked on top of each other. See example of how to convert to wide format. This can be produced using extract_sigma().
isotope_a	character string that is the column name of the first isotope used in dat_sigma. Defaults to "d13c".
isotope_b	character string that is the column name of the second isotope used in dat_sigma. Defaults to "d15n".
p_ell	is the confidence interval of each ellipse estimate. Default is 0.95 (i.e., 95% confidence interval). This value is bound by 0 and 1 and has to be a numeric.
random	logical value indicating whether or not to randomly sample posterior distributions for μ and Σ to create a sub-sample of ellipse. Default is TRUE.
set_seed	numerical value to set seed for random sampling. Default is a random value. To consistently sample the same subsample, please supply a numerical value (e.g., 4). It is highly suggested to use set_seed to make the function results when randomly sampling reproducible.
n	numerical value that controls the number of random samples. Default is 10.

message	control whether the time processing is displayed after the end of the function. Default is TRUE.
...	additional paramaters to pass to ellipse::ellipse(). See that function for more detail.

Value

A tibble containing, sample_name, sample_number, and the isotopes that were used in the estimation of ellipse (i.e., and d13c and d15n).

See Also

nicheROVER::niw.post(), SIBER::siberMVN(), extract_mu(), and extract_sigma()

Examples

```
niche_ellipse(dat_mu = mu_est_long,
              dat_sigma = sigma_est_wide)
```

---

niw_fish_post *A* list *of the posterior estimates of* μ *and* Σ *from* {nicheROVER}

Description

Posterior estimates of μ and Σ using the fish data set from nicheROVER, using Normal-Inverse-Wishart (NIW) priors. This list is produced using the function niw.post() from nicheROVER.

Usage

```
niw_fish_post
```

Format

A list with elements μ and Σ of sizes c(nsamples, length(lambda)) and c(dim(Psi)).

`over_stat` *A* `data.frame` *containing the estimates of percentage of overlap among groups*

Description

Estimates of the percentage of overlap among example species used in nicheROVER.

Usage

```
over_stat
```

Format

A `array`containing `matrices` of the percent overlap for each group used in Bayesian estimates of μ and Σ using Normal-Inverse-Wishart (NIW) priors calculated in `niw.post()`.

`post_sam_siber` *A* `list` *of the posterior estimates of* μ *and* Σ *from* {SIBER}

Description

Posterior estimates of μ and Σ using the `demo.siber.data.2` data set from SIBER. This `list` is produced using the function `siberMVN()` from SIBER.

Usage

```
post_sam_siber
```

Format

A `list` with estimates of μ and Σ for each species and group.

`sigma_est_wide` *A* `data.frame` *containing posterior estimates of* Σ

Description

Posterior estimates of Σ using `fish` data set from nicheROVER, using Normal-Inverse-Wishart (NIW) priors

Usage

```
sigma_est_wide
```

Format

`data.frame` containing 8,000 rows and 6 variables

metric name of the metric extracted from `niw.post()`

species species abbreviation

isotope column with isotope name

sample_number sample number from 1-1000

d15n estimate of Σ for d15n produced from `niw.post()`

d13c estimate of Σ for d13c produced from `niw.post()`

Index

∗ **datasets**
mu_est_long, 13
niw_fish_post, 15
over_stat, 16
post_sam_siber, 16
sigma_est_wide, 17

create_comparisons, 2

extract_group_metrics, 3
extract_layman, 4
extract_mu, 7
extract_mu(), *15*
extract_niche_size, 8
extract_overlap, 10
extract_sigma, 10
extract_sigma(), *15*
extract_similarities, 12

mu_est_long, 13

niche_ellipse, 14
nicheROVER::niche.size(), *9*
nicheROVER::niw.post(), *8*, *9*, *11*, *15*
nicheROVER::overlap(), *10*
niw_fish_post, 15

over_stat, 16

post_sam_siber, 16

SIBER::bayesianLayman(), *6*
SIBER::createSiberObject(), *6*, *9*
SIBER::siberEllipses(), *9*
SIBER::siberMVN(), *8*, *11*, *15*
sigma_est_wide, 17