Package 'estar'

November 28, 2025

Title Ecological Stability Metrics

Version 1.0-1

Description Standardises and facilitates the use of eleven established stability properties that have been used to assess systems' responses to press or pulse disturbances at different ecological levels (e.g. population, community). There are two sets of functions. The first set corresponds to functions that measure stability at any level of organisation, from individual to community and can be applied to a time series of a system's state variables (e.g., body mass, population abundance, or species diversity). The properties included in this set are: invariability, resistance, extent and rate of recovery, persistence, and overall ecological vulnerability. The second set of functions can be applied to Jacobian matrices. The functions in this set measure the stability of a community at short and long time scales. In the short term, the community's response is measured by maximal amplification, reactivity and initial resilience (i.e. initial rate of return to equilibrium). In the long term, stability can be measured as asymptotic resilience and intrinsic stochastic invariability. Figueiredo et al. (2025) <doi:10.32942/X2M053>.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Suggests testthat (>= 3.0.0), knitr, rmarkdown, Matrix, hesim, cowplot, viridis, forcats, tidyr, utils, MARSS, magrittr, dplyr, purrr, tibble, kableExtra

Depends R (>= 4.1.0)

Imports ggplot2, stats, vegan, zoo

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Ludmilla Figueiredo [aut, cre] (ORCID: <https://orcid.org/0000-0001-8217-7800>), Viktoriia Radchuk [aut] (ORCID: <https://orcid.org/0000-0003-3072-0095>), Cédric Scherer [ctb] (ORCID: <https://orcid.org/0000-0003-0465-2543>)

Maintainer Ludmilla Figueiredo <ludmilla.figueiredo@protonmail.com>

Repository CRAN

Date/Publication 2025-11-28 13:20:24 UTC

Contents

`aquacomm_fgps` *Macroinvertebrate aquatic community.*

Description

Dataset collected in an ecotoxicological study about the effects of insecticide (chlorpyrifos) use on a macroinvertebrate aquatic community (van den Brink et a. 1996, Wijngaarden_effects_1996). The community is composed of 128 species, classified into 5 functional groups: herbivores, detri-herbivores, carnivores, omnivores, and detrivores.

Usage

```
aquacomm_fgps
```

Format

A data frame with five variables:

Fields

`time` sequential week number relative to the application of insecticide

`treat` concentration of insecticide

`repli` number of replicate

`herb` abundance of herbivores

`detr_herb` abundance of detri-herbivores

`carn` abundance of carnivores

`omni` abundance of omnivores

`detr` abundance of detrivores

References

van den Brink, P. J., Van Wijngaarden, R. P. A., Lucassen, W. G. H., Brock, T. C. M., & Leeuwangh, P. (1996). Environmental Toxicology and Chemistry, 15(7), 1143–1153. doi:10.1002/etc.5620150719

Wijngaarden, R. P. A. van, Brink, P. J. van den, Crum, S. J. H., Brock, T. C. M., Leeuwangh, P., & Voshaar, O. J. H. (1996). Effects of the insecticide dursban® 4E (active ingredient chlorpyrifos) in outdoor experimental ditches: I. Comparison of short-term toxicity between the laboratory and the field. Environmental Toxicology and Chemistry, 15(7), 1133–1142. doi:10.1002/etc.5620150718

`aquacomm_resps` *Macroinvertebrate aquatic community formatted for univariate metrics.*

Description

Data frame created from `aquacomm_fgps`

Usage

```
aquacomm_resps
```

Format

A data frame with five variables:

Fields

`time` sequential week number relative to the application of insecticide

`carn_0` mean abundance of carnivores in control replicates

`carn_0.1` mean abundance of carnivores in replicates subjected to pulse application of 0.1 nano g/L of chlorpyrifos insecticide

`carn_0.9` mean abundance of carnivores in replicates subjected to 0.9 nano g/L

`carn_6` mean abundance of carnivores in replicates subjected to 6 nano g/L

`carn_44` mean abundance of carnivores in replicates subjected to 44 nano g/L

Source

```
aquacomm_resps <- aquacomm_fgps |> dplyr::select(-c(herb, detr_herb, omni, detr)) |>
dplyr::group_by(time, treat) |> dplyr::summarize_at("carn", mean) |> dplyr::ungroup()
|> tidyr::pivot_wider(names_from = treat, values_from = carn, names_prefix = "carn_")
|> dplyr::select(time, "statvar_bl" = carn_0, "statvar_db" = carn_6) usethis::use_data(aquacomm_resps)
```

`asympt_resil` *Calculate the asymptotic resilience of a community after disturbance*

Description

`asympt_resil` calculates a community's asymptotic resilience R_∞ as the slowest long-term asymptotic rate of return to equilibrium after a pulse perturbation (Arnoldi et al. 2016, Downing et al. 2020).

Usage

```
asympt_resil(B)
```

Arguments

`B`	a matrix, containing the interactions between the species or functional groups in the community. Can be calculated with `extractB` from the fitted MARSS object.

Details

$$R_\infty = -\log(|\lambda_{\mathrm{dom}}(B)|)$$

Value

A single positive numeric value, the asymptotic rate of return to equilibrium after a pulse perturbation.The larger its value, the more stable the system.

References

Arnoldi et al. (2016). Resilience, reactivity and variability: A mathematical comparison of ecological stability measures. Journal of Theoretical Biology, 389, 47–59. doi:10.1016/j.jtbi.2015.10.012

Downing, A. L., Jackson, C., Plunkett, C., Lockhart, J. A., Schlater, S. M., & Leibold, M. A. (2020). Temporal stability vs. Community matrix measures of stability and the role of weak interactions. Ecology Letters, 23(10), 1468–1478. doi:10.1111/ele.13538

See Also

`extractB()`

Examples

```
library(MARSS)


 # smaller dataset for example:
 # 3 functional groups and two insecticide concentrations besides control
 data_df <- subset(
   aquacomm_fgps,
   treat %in% c(0.0, 0.9, 6) &
     time >= 1 & time <= 28,
   select = c(time, treat, repli, herb, carn, detr)
 )

 # estimate z-score transformation and replace zeros with NA
 data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                MARSS::zscore)
 data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                function(x) replace(x, x == 0, NA))

 # reshape data from wide to long format
 data_z_ldf <- reshape(
   data_df,
   varying = list(c("herb", "carn", "detr")),
   v.names = "abund_z",
   timevar = "fgp",
   times = c("herb", "carn", "detr"),
   direction = "long",
   idvar = c("time", "treat", "repli")
 )

 data_z_ldf <- data_z_ldf[order(data_z_ldf$time, data_z_ldf$treat, data_z_ldf$fgp),]

 # summarize mean and sd
 data_z_summldf <- aggregate(abund_z ~ time + treat + fgp, data_z_ldf,
                             function(x) c(mean = mean(x, na.rm = TRUE),
                                           sd = sd(x, na.rm = TRUE)))
 data_z_summldf <- do.call(data.frame, data_z_summldf)
 names(data_z_summldf)[4:5] <- c("abundz_mu", "abundz_sd")

 # split dataframe per functional groups
 # into list to apply the MARSS model more easily
split_data_z <- split(data_z_summldf[, c("time", "fgp", "abundz_mu")], data_z_summldf$treat)

 reshape_to_wide <- function(df) {
   df_wide <- reshape(df,
                      idvar = "fgp",
                      timevar = "time",
                      direction = "wide")
   rownames(df_wide) <- df_wide$fgp
   df_wide <- df_wide[, -1]  # Remove the 'fgp' column
   as.matrix(df_wide)
 }
```

```
data_z_summls <- lapply(split_data_z, reshape_to_wide)

# fit MARSS models
data.marssls <- list(
  MARSS(
    data_z_summls[[1]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[2]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[3]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  )
)

# identify experiments
names(data.marssls) <- paste0("Conc. = ", c("0", "0.9", "44"), " micro g/L")

# extract community matrices (B)
data.Bls <- data.marssls |>
  lapply(extractB,
         states_names = c("Herbivores", "Carnivores", "Detrivores"))
```

```
# calculate asymptotic resilience for each of the B matrices
purrr::map(data.Bls, asympt_resil)
```

`common_params` *Define parameters that are common to all functions*

Description

Define parameters that are common to all functions

Usage

```
common_params(
  type,
  response,
  vd_i,
  td_i,
  d_data,
  vb_i,
  tb_i,
  b_data,
  b,
  b_tf,
  na_rm,
  metric_tf,
  comm_d,
  comm_b,
  comm_t,
  method,
  binary
)
```

Arguments

`type`	a string defining the type of stability (`"functional"` or `"compositional"`) to be calculated.
`response`	a string stating whether the stability metric should be calculated using the log-response ratio between the values in the disturbed system and the baseline (`response = "lrr"`) or using the state variable values in the disturbed system alone (`response == "v"`).
`vd_i`	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in `d_data`.
`td_i`	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in `d_data`.

d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in b_data containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in b_data containing said the time variable in the dataframe with baseline values.
b_data	an optional data frame containing the time series of the state variable values in the baseline.
b	a string stating whether the baseline is defined by a separate baseline that is specified by the user (b = "input") or by a period of the disturbed system (b = "d") prior to the disturbance. This period is specified by b_tf.
b_tf	a numerical vector, specifying the beginning and end of the pre-disturbance time period for the disturbed time-series that defines the baseline. Obligatory if (b = "d"), see 'Details'.
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.
metric_tf	a numerical vector, specifying the beginning and end of the time period over which the stability metric should be measured.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_t	the name of the time variable in comm_b and comm_d.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".

---

comm_base *Macroinvertebrate aquatic community data formatted for calculation of compositional stability metrics*

Description

Data frame created with data-raw/estar_data.R. Data for the baseline community: mean abundance calculated from the values for the control experiments one and 42 days after application of the insecticide.

Usage

```
comm_base
```

Format

A data frame with five variables:

Fields

`time` sequential week number relative to the application of insecticide

`herb` mean abundance of herbivores

`detr_herb` mean abundance of detri-herbivores

`carn` mean abundance of carnivores

`omni` mean abundance of omnivores

`detr` mean abundance of detrivores

---

comm_dist *Macroinvertebrate aquatic community data formatted for calculation of compositional stability metrics*

Description

Data frame created with `data-raw/estar_data.R`. Data for a disturbed community: mean abundance calculated from the values for the experiments under treatment with 6 microg / L of insecticide.

Usage

```
comm_dist
```

Format

A data frame with five variables:

Fields

`time` sequential week number relative to the application of insecticide

`herb` mean abundance of herbivores

`detr_herb` mean abundance of detri-herbivores

`carn` mean abundance of carnivores

`omni` mean abundance of omnivores

`detr` mean abundance of detrivores

`extractB` *Extract the community matrix (B)*

Description

Extract the community matrix (B) estimated by a MARSS model. The matrix is necessary to calculate the Jacobian metrics.

Usage

```
extractB(marss_res, states_names = NULL)
```

Arguments

`marss_res`	MARSS object returned by `MARSS`
`states_names`	a string vector containing the names of species/groups for which interactions were estimated.

Value

A named square matrix (B) with one row (and column) per species/group listed in `states_names` (which should also have been used in `MARSS`). Row and column names are included (named after `states_names`). The values are interaction strengths between the species/groups estimated by MARSS.

References

Holmes, E. E., Ward, E. J., Scheuerell, M. D., & Wills, K. (2024). MARSS: Multivariate Autoregressive State-Space Modeling (Version 3.11.9). doi:10.32614/CRAN.package.MARSS

Holmes, E. E., Scheuerell, M. D., & Ward, E. J. (2024). Analysis of multivariate time-series using the MARSS package. Version 3.11.9. doi:10.5281/zenodo.5781847

Holmes, E. E., Ward, E. J., & Wills, K. (2012). MARSS: Multivariate autoregressive state-space models for analyzing time-series data. The R Journal, 4(1), 30. doi:10.32614/RJ2012002

Examples

```
library(MARSS)


  # smaller dataset for example:
  # 3 functional groups and two insecticide concentrations besides control
  data_df <- subset(
    aquacomm_fgps,
    treat %in% c(0.0, 0.9, 6) &
      time >= 1 & time <= 28,
    select = c(time, treat, repli, herb, carn, detr)
  )
```

```
# estimate z-score transformation and replace zeros with NA
data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                               MARSS::zscore)
data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                               function(x) replace(x, x == 0, NA))

# reshape data from wide to long format
data_z_ldf <- reshape(
  data_df,
  varying = list(c("herb", "carn", "detr")),
  v.names = "abund_z",
  timevar = "fgp",
  times = c("herb", "carn", "detr"),
  direction = "long",
  idvar = c("time", "treat", "repli")
)

data_z_ldf <- data_z_ldf[order(data_z_ldf$time, data_z_ldf$treat, data_z_ldf$fgp),]

# summarize mean and sd
data_z_summldf <- aggregate(abund_z ~ time + treat + fgp, data_z_ldf, function(x)
  c(mean = mean(x, na.rm = TRUE), sd = sd(x, na.rm = TRUE)))
data_z_summldf <- do.call(data.frame, data_z_summldf)
names(data_z_summldf)[4:5] <- c("abundz_mu", "abundz_sd")

# split dataframe per functional groups
# into list to apply the MARSS model more easily
split_data_z <- split(data_z_summldf[, c("time", "fgp", "abundz_mu")], data_z_summldf$treat)

reshape_to_wide <- function(df) {
  df_wide <- reshape(df,
                     idvar = "fgp",
                     timevar = "time",
                     direction = "wide")
  rownames(df_wide) <- df_wide$fgp
  df_wide <- df_wide[, -1]  # Remove the 'fgp' column
  as.matrix(df_wide)
}

data_z_summls <- lapply(split_data_z, reshape_to_wide)

# fit MARSS models
data.marssls <- list(
  MARSS(
    data_z_summls[[1]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
```

```
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[2]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[3]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  )
)

# identify experiments
names(data.marssls) <- paste0("Conc. = ", c("0", "0.9", "44"), " micro g/L")

# extract community matrices (B)
data.Bls <- data.marssls |>
  lapply(extractB,
         states_names = c("Herbivores", "Carnivores", "Detrivores"))

print(data.Bls)
```

`init_resil` *Calculate the initial resilience of a community after disturbance*

Description

`init_resil` calculates initial resilience R_0 as the initial rate of return to equilibrium (Downing et al. 2020).

Usage

```
init_resil(B)
```

Arguments

B	a matrix, containing the interactions between the species or functional groups in the community. Can be calculated with extractB from the fitted MARSS object.

Details

$$R_0 = -\log\left(\sqrt{\lambda_{\mathrm{dom}}(B^{\mathsf{T}}B)}\right)$$

where λ_{dom} is the dominant eigenvalue.

Value

A single numeric value, the initial resilience. The larger its value, the more stable the system.

References

Downing, A. L., Jackson, C., Plunkett, C., Lockhart, J. A., Schlater, S. M., & Leibold, M. A. (2020). Temporal stability vs. Community matrix measures of stability and the role of weak interactions. Ecology Letters, 23(10), 1468–1478. doi:10.1111/ele.13538

See Also

extractB()

Examples

```
library(MARSS)


  # smaller dataset for example:
  # 3 functional groups and two insecticide concentrations besides control
  data_df <- subset(
    aquacomm_fgps,
    treat %in% c(0.0, 0.9, 6) &
      time >= 1 & time <= 28,
    select = c(time, treat, repli, herb, carn, detr)
  )

  # estimate z-score transformation and replace zeros with NA
  data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                 MARSS::zscore)
  data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                 function(x) replace(x, x == 0, NA))

  # reshape data from wide to long format
```

```
data_z_ldf <- reshape(
  data_df,
  varying = list(c("herb", "carn", "detr")),
  v.names = "abund_z",
  timevar = "fgp",
  times = c("herb", "carn", "detr"),
  direction = "long",
  idvar = c("time", "treat", "repli")
)

data_z_ldf <- data_z_ldf[order(data_z_ldf$time, data_z_ldf$treat, data_z_ldf$fgp), ]

# summarize mean and sd
data_z_summldf <- aggregate(abund_z ~ time + treat + fgp, data_z_ldf, function(x)
  c(mean = mean(x, na.rm = TRUE), sd = sd(x, na.rm = TRUE)))
data_z_summldf <- do.call(data.frame, data_z_summldf)
names(data_z_summldf)[4:5] <- c("abundz_mu", "abundz_sd")

# split dataframe per functional groups
# into list to apply the MARSS model more easily
split_data_z <- split(data_z_summldf[, c("time", "fgp", "abundz_mu")], data_z_summldf$treat)

reshape_to_wide <- function(df) {
  df_wide <- reshape(df,
                     idvar = "fgp",
                     timevar = "time",
                     direction = "wide")
  rownames(df_wide) <- df_wide$fgp
  df_wide <- df_wide[, -1]  # Remove the 'fgp' column
  as.matrix(df_wide)
}

data_z_summls <- lapply(split_data_z, reshape_to_wide)

# fit MARSS models
data.marssls <- list(
  MARSS(
    data_z_summls[[1]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[2]],
    model = list(
      B = "unconstrained",
```

```
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[3]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  )
)

# identify experiments
names(data.marssls) <- paste0("Conc. = ", c("0", "0.9", "44"), " micro g/L")

# extract community matrices (B)
data.Bls <- data.marssls |>
  lapply(extractB,
         states_names = c("Herbivores", "Carnivores", "Detrivores"))

# calculate initial resilience for each of the B matrices
purrr::map(data.Bls, init_resil)
```

`invariability` *Calculate the invariability of a state variable after disturbance*

Description

`invariability` returns the temporal invariability I of a system following disturbance. Invariability can be calculated using the post-disturbance values of the state variable in the disturbed system as the response, or the log-response ratio of the state variable in the disturbed system compared to the baseline. Two variants of invariability can be calculated: as the inverse of the coefficient of variation of the system's response, or the inverse of the standard deviation of residuals of the linear model that uses the time as the predictor of the system's response.

Usage

```
invariability(
  type,
  mode = NULL,
  response = NULL,
  metric_tf,
  vd_i = NULL,
  td_i = NULL,
  d_data = NULL,
  vb_i = NULL,
  tb_i = NULL,
  b_data = NULL,
  comm_b = NULL,
  comm_d = NULL,
  comm_t = NULL,
  method = "bray",
  binary = "FALSE",
  na_rm = TRUE
)
```

Arguments

type	a string defining the type of stability (`"functional"` or `"compositional"`) to be calculated.
mode	A string stating which variant of invariability should be calculated, the one based on the coefficient of variation of the state variable `mode = "cv"`, or the one based on fitting the linear model `"lm_res"`.
response	a string stating whether the stability metric should be calculated using the log-response ratio between the values in the disturbed system and the baseline (`response = "lrr"`) or using the state variable values in the disturbed system alone (`response == "v"`).
metric_tf	a numerical vector, specifying the beginning and end of the time period over which the stability metric should be measured.
vd_i	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in `d_data`.
td_i	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in `d_data`.
d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in `b_data` containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in `b_data` containing said the time variable in the dataframe with baseline values.

b_data	an optional data frame containing the time series of the state variable values in the baseline.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_t	the name of the time variable in comm_b and comm_d.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.

Details

Instability can be calculated as the coefficient fo variation of the system:

- For functional stability, the response is the log response ratio between the state variable's value in the disturbed (v_d) and in the baseline systems (v_b or v_p if the baseline is pre-disturbance values) or the state variable's value in the disturbed system itself. Therefore, $I = \mathrm{CV}\left(\log\left(\frac{v_d}{v_b}\right)\right)^{-1}$, or $I = \mathrm{CV}\left(\log\left(\frac{v_d}{v_p}\right)\right)^{-1}$, or $I = \mathrm{CV}(v_d)^{-1}$.
- For compositional stability, the response is the dissimilarity between the disturbed (C_d) and baseline (C_b) communities: $I = \mathrm{CV}\left(\mathrm{dissim}\left(\frac{C_d}{C_b}\right)\right)^{-1}$

Alternatively, instability can be calculated as inverse of the standard deviation of residuals of the linear model where the response (same as above) is predicted by time, whereby: $I = \sigma(\varepsilon)^{-1}$, from $y = \alpha + R_r\, t + \varepsilon$, where $y \in \left\{\log\left(\frac{v_d}{v_b}\right), \log\left(\frac{v_d}{v_p}\right), v_d, \mathrm{dissim}\left(\frac{C_d}{C_b}\right)\right\}$, α is the intercept, and R_r is the recovery rate.

Value

A single numeric, the invariability (I) value. The larger in magnitude I is, the higher the stability, since the variation around the trend is lower.

Examples

```
invariability(
  vd_i = "statvar_db", td_i = "time", response = "v", mode = "cv",
  metric_tf = c(11, 50), d_data = aquacomm_resps, type = "functional"
)
invariability(
  vd_i = aquacomm_resps$statvar_db, td_i = aquacomm_resps$time,
  response = "v", mode = "cv", metric_tf = c(11, 50), type = "functional"
)
invariability(
```

```
  vd_i = "statvar_db", td_i = "time", response = "lrr", mode = "lm_res",
  metric_tf = c(11, 50), d_data = aquacomm_resps, vb_i = "statvar_bl",
  tb_i = "time", b_data = aquacomm_resps, type = "functional"
)
invariability(
  vd_i = aquacomm_resps$statvar_db, td_i = aquacomm_resps$time,
  response = "lrr", type = "functional",
  metric_tf = c(11, 50), mode = "lm_res", vb_i = aquacomm_resps$statvar_bl,
  tb_i = aquacomm_resps$time
)
invariability(
  type = "compositional", metric_tf = c(0.14, 56), comm_d = comm_dist,
  comm_b = comm_base, comm_t = "time"
)
```

`max_amp` *Calculate the maximal amplification of a community after disturbance*

Description

`max_amp` calculates the maximal amplification (A_{max}) as the euclidean norm of a community matrix B (Neubert et al. 1996). It uses the `expmat` function of the `hesim` package to calculate the exponential of the community matrix B , and then its Euclidean norm.

Usage

```
max_amp(B)
```

Arguments

`B`	a matrix, containing the interactions between the species or functional groups in the community. Can be calculated with `extractB` from the fitted MARSS object.

Details

$$A_{max} = \max_{t \geq 0} \left(\max_{x_0 \neq 0} \frac{\left\| e^{B^{\mathsf{T}} t} x_0 \right\|}{\|x_0\|} \right)$$

Value

A single numeric numeric, the maximal amplification factor by which a perturbation is amplified (relative to its initial size) before the system's eventual equilibrium. If $A_{max} > 1$, the system overreacts and departs from equilibrium. If $A_{max} \approx 1$, the system is stable.

References

Neubert, M. G., & Caswell, H. (1997). Alternatives to Resilience for Measuring the Responses of Ecological Systems to Perturbations. Ecology, 78(3), 653-665.

Devin Incerti and Jeroen P Jansen (2021). hesim: Health Economic Simulation Modeling and Decision Analysis. URL: doi:10.48550/arXiv.2102.09437

See Also

extractB()

Examples

```
library(MARSS)


  # smaller dataset for example:
  # 3 functional groups and two insecticide concentrations besides control
  data_df <- subset(
    aquacomm_fgps,
    treat %in% c(0.0, 0.9, 6) &
      time >= 1 & time <= 28,
    select = c(time, treat, repli, herb, carn, detr)
  )

  # estimate z-score transformation and replace zeros with NA
  data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                 MARSS::zscore)
  data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                 function(x) replace(x, x == 0, NA))

  # reshape data from wide to long format
  data_z_ldf <- reshape(
    data_df,
    varying = list(c("herb", "carn", "detr")),
    v.names = "abund_z",
    timevar = "fgp",
    times = c("herb", "carn", "detr"),
    direction = "long",
    idvar = c("time", "treat", "repli")
  )

  data_z_ldf <- data_z_ldf[order(data_z_ldf$time, data_z_ldf$treat, data_z_ldf$fgp),]

  # summarize mean and sd
  data_z_summldf <- aggregate(abund_z ~ time + treat + fgp, data_z_ldf, function(x)
    c(mean = mean(x, na.rm = TRUE), sd = sd(x, na.rm = TRUE)))
  data_z_summldf <- do.call(data.frame, data_z_summldf)
  names(data_z_summldf)[4:5] <- c("abundz_mu", "abundz_sd")

  # split dataframe per functional groups
  # into list to apply the MARSS model more easily
```

```
split_data_z <- split(data_z_summldf[, c("time", "fgp", "abundz_mu")], data_z_summldf$treat)

reshape_to_wide <- function(df) {
  df_wide <- reshape(df,
                     idvar = "fgp",
                     timevar = "time",
                     direction = "wide")
  rownames(df_wide) <- df_wide$fgp
  df_wide <- df_wide[, -1]  # Remove the 'fgp' column
  as.matrix(df_wide)
}

data_z_summls <- lapply(split_data_z, reshape_to_wide)

# fit MARSS models
data.marssls <- list(
  MARSS(
    data_z_summls[[1]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[2]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[3]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
```

```
    method = "BFGS"
  )
)

# identify experiments
names(data.marssls) <- paste0("Conc. = ", c("0", "0.9", "44"), " micro g/L")

# extract community matrices (B)
data.Bls <- data.marssls |>
  lapply(extractB,
         states_names = c("Herbivores", "Carnivores", "Detrivores"))

# calculate maximal amplification for each of the B matrices
purrr::map(data.Bls, max_amp)
```

---

oev *Calculate the overall ecological vulnerability of a community after disturbance*

Description

oev returns the overall ecological vulnerability OEV , calculated as the area under the curve (AUC) of the absolute log-response-ratio (functional stability) or the dissimilarity (compositional stability) between the disturbed and baseline communities . The area under the curve is calculated through trapezoidal integration.

Usage

```
oev(
  type,
  metric_tf,
  response,
  vd_i,
  td_i,
  d_data = NULL,
  vb_i = NULL,
  tb_i = NULL,
  b_data = NULL,
  comm_d = NULL,
  comm_b = NULL,
  comm_t = NULL,
  method = "bray",
  binary = "FALSE",
  na_rm = TRUE
)
```

Arguments

type	a string defining the type of stability ("functional" or "compositional") to be calculated.
metric_tf	a numerical vector, specifying the beginning and end of the time period over which the stability metric should be measured.
response	a string stating whether the stability metric should be calculated using the log-response ratio between the values in the disturbed system and the baseline (response = "lrr") or using the state variable values in the disturbed system alone (response == "v").
vd_i	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in d_data.
td_i	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in d_data.
d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in b_data containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in b_data containing said the time variable in the dataframe with baseline values.
b_data	an optional data frame containing the time series of the state variable values in the baseline.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_t	the name of the time variable in comm_b and comm_d.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.

Details

The overall ecosystem variability (OEV) is defined as the area under the curve (AUC) of the system's response through time. For functional stability, the response is the log response ratio between the state variable's value in the disturbed (v_d) and in the baseline systems (v_b or v_p if the baseline is pre-disturbance values). For compositional stability, the response is the dissimilarity between the disturbed (C_d) and baseline (C_b) communities. Therefore,

$$\text{OEV} = \text{AUC}\left(\left|\log\left(\frac{v_d(t)}{v_b(t)}\right)\right|, t\right)$$

or

$$\text{OEV} = \text{AUC}\left(\left|\log\left(\frac{v_d(t)}{v_p(t)}\right)\right|, t\right)$$

or

$$\text{OEV} = \text{AUC}\left(\text{dissim}\left(\frac{C_d(t)}{C_b(t)}\right), t\right)$$

where the area under the curve is defined as

$\text{AUC}(y,t) = \sum_{i=1}^{n-1} \frac{(t_{i+1}-t_i)(y_i+y_{i+1})}{2}$ (trapezoidal integration).

Value

A single numeric value, the overall ecological vulnerability. $OEV \geq 0$. The higher the value, the less stable the system.

References

Urrutia-Cordero, P., Langenheder, S., Striebel, M., Angeler, D. G., Bertilsson, S., Eklöv, P., Hansson, L.-A., Kelpsiene, E., Laudon, H., Lundgren, M., Parkefelt, L., Donohue, I., & Hillebrand, H. (2022). Integrating multiple dimensions of ecological stability into a vulnerability framework. Journal of Ecology, 110(2), 374–386. doi:10.1111/13652745.13804

Examples

```
oev(
  type = "functional", response = "lrr", metric_tf = c(0.14, 56), vd_i = "statvar_db",
  td_i = "time", d_data = aquacomm_resps, vb_i = "statvar_bl", tb_i = "time",
  b_data = aquacomm_resps
)
oev(
  type = "compositional", metric_tf = c(0.14, 56), comm_d = comm_dist,
  comm_b = comm_base, comm_t = "time"
)
```

`persistence` *Calculate the persistence of a state variable over a defined time interval*

Description

`persistence` (P) returns the proportion of time the state variable remained inside the interval defined by one baseline's $\pm$ sd from the baseline's mean (functional stability) or the user-defined upper and lower limits of the community dissimilarity (compositional stability). The proportion is calculated in relation to the time period (`metric_tf`) defined by the user.

Usage

```
persistence(
  type,
  metric_tf,
  b,
  b_tf = NULL,
  vd_i,
  td_i,
  d_data = NULL,
  vb_i = NULL,
  tb_i = NULL,
  b_data = NULL,
  comm_d = NULL,
  comm_b = NULL,
  comm_t = NULL,
  method = "bray",
  binary = "FALSE",
  low_lim = NULL,
  high_lim = NULL,
  na_rm = TRUE
)
```

Arguments

type	a string defining the type of stability ("functional" or "compositional") to be calculated.
metric_tf	a numerical vector, specifying the beginning and end of the time period over which the stability metric should be measured.
b	a string stating whether the baseline is defined by a separate baseline that is specified by the user (b = "input") or by a period of the disturbed system (b = "d") prior to the disturbance. This period is specified by b_tf.
b_tf	a numerical vector, specifying the beginning and end of the pre-disturbance time period for the disturbed time-series that defines the baseline. Obligatory if (b = "d"), see 'Details'.
vd_i	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in d_data.
td_i	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in d_data.
d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in b_data containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in b_data containing said the time variable in the dataframe with baseline values.

b_data	an optional data frame containing the time series of the state variable values in the baseline.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_t	the name of the time variable in comm_b and comm_d.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".
low_lim	minimal dissimilarity value the user expects for a persistent community
high_lim	maximal dissimilarity value the user expects for a persistent community
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.

Details

$$P = \frac{t_P}{t_a}$$

where t_a is the total time frame defined by the user (metric_tf) and t_P is the time period during which the response remain inside the limits defined by the user.

If the baseline is defined by the pre-disturbed values of the state variable in the disturbed system (b = "d"), this pre-disturbed time period used as baseline (b_tf) cannot overlap with the time period for which the persistence is to be calculated (metric_tf), because of redundancy: the values over b_tf define the interval for which the values in metric_tf are checked. If they are the same (or partly, if overlap is partial), the returned value of persistence will be falsely higher.

Value

A single numeric value, the persistence, $0 \leq P \leq 1$. The higher persistence is, the more stable the system.

Examples

```
persistence(
  type = "functional", vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "d",
  b_tf = c(1, 9), metric_tf = c(28, 56)
)
persistence(
  type = "functional", vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "d",
  b_tf = c(1, 9), metric_tf = c(28, 56)
)
persistence(
  type = "functional", vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "input",
```

```
  metric_tf = c(28, 56), vb_i = "statvar_bl", tb_i = "time",
  b_data = aquacomm_resps
)
persistence(
 type = "functional", vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "input",
  metric_tf = c(28, 56), vb_i = "statvar_bl", tb_i = "time",
  b_data = aquacomm_resps
)
persistence(
  type = "compositional", b = "input",metric_tf = c(28, 56), comm_d = comm_dist,
  comm_b = comm_base, comm_t = "time", low_lim = 0.5, high_lim = 0.9
)
```

`reactivity` *Calculate the reactivity of a community after disturbance*

Description

`reactitvity` calculates reactivity R_a as the initial rate of return to equilibrium (Downing et al. 2020). Following Neubert et al. (1996), it is calculated as the largest eigenvalue of the Hermitian part of the community matrix B .

Usage

```
reactivity(B)
```

Arguments

`B`	a matrix, containing the interactions between the species or functional groups in the community. Can be calculated with `extractB` from the fitted MARSS object.

Details

$R_a = \lambda_{\mathrm{dom}}(H(B))$ where λ_{dom} is the dominant eigenvalue, and $H(B)$ is the Rayleigh quotient of $B : H(B) = \frac{x^{\mathsf{T}} B\, x}{x^{\mathsf{T}} x}$.

Value

A single numeric value, the reactivity value. If $R_a > 0$, the perturbation grows, i.e., the system is initially destabilised. If $R_a < 0$ 0 the perturbation doesn't grow and the system isn't initially destabilised.

References

Neubert, M. G., & Caswell, H. (1997). Alternatives to Resilience for Measuring the Responses of Ecological Systems to Perturbations. Ecology, 78(3), 653–665.

Downing, A. L., Jackson, C., Plunkett, C., Lockhart, J. A., Schlater, S. M., & Leibold, M. A. (2020). Temporal stability vs. Community matrix measures of stability and the role of weak interactions. Ecology Letters, 23(10), 1468–1478. doi:10.1111/ele.13538

See Also

extractB()

Examples

```
library(MARSS)


  # smaller dataset for example:
  # 3 functional groups and two insecticide concentrations besides control
  data_df <- subset(
    aquacomm_fgps,
    treat %in% c(0.0, 0.9, 6) &
      time >= 1 & time <= 28,
    select = c(time, treat, repli, herb, carn, detr)
  )

  # estimate z-score transformation and replace zeros with NA
  data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                 MARSS::zscore)
  data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                                 function(x) replace(x, x == 0, NA))

  # reshape data from wide to long format
  data_z_ldf <- reshape(
    data_df,
    varying = list(c("herb", "carn", "detr")),
    v.names = "abund_z",
    timevar = "fgp",
    times = c("herb", "carn", "detr"),
    direction = "long",
    idvar = c("time", "treat", "repli")
  )

  data_z_ldf <- data_z_ldf[order(data_z_ldf$time, data_z_ldf$treat, data_z_ldf$fgp), ]

  # summarize mean and sd
  data_z_summldf <- aggregate(abund_z ~ time + treat + fgp, data_z_ldf, function(x)
    c(mean = mean(x, na.rm = TRUE), sd = sd(x, na.rm = TRUE)))
  data_z_summldf <- do.call(data.frame, data_z_summldf)
  names(data_z_summldf)[4:5] <- c("abundz_mu", "abundz_sd")

  # split dataframe per functional groups
  # into list to apply the MARSS model more easily
 split_data_z <- split(data_z_summldf[, c("time", "fgp", "abundz_mu")], data_z_summldf$treat)

  reshape_to_wide <- function(df) {
    df_wide <- reshape(df,
                       idvar = "fgp",
                       timevar = "time",
                       direction = "wide")
    rownames(df_wide) <- df_wide$fgp
```

```
  df_wide <- df_wide[, -1]  # Remove the 'fgp' column
  as.matrix(df_wide)
}

data_z_summls <- lapply(split_data_z, reshape_to_wide)

# fit MARSS models
data.marssls <- list(
  MARSS(
    data_z_summls[[1]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[2]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[3]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  )
)

# identify experiments
names(data.marssls) <- paste0("Conc. = ", c("0", "0.9", "44"), " micro g/L")

# extract community matrices (B)
```

```
data.Bls <- data.marssls |>
  lapply(extractB,
         states_names = c("Herbivores", "Carnivores", "Detrivores"))

# calculate reactivity for each of the B matrices
purrr::map(data.Bls, reactivity)
```

`recovery_extent` *Calculate the extent of recovery after disturbance*

Description

`recovery_extent` (R_e) calculates how close a state variable is to its baseline value at a time point specified by the user (usually after recovery has taken place). For functional stability, the distance can be calculated as the log-response ratio or as the difference between the state variables in a disturbed time-series and the baseline. The baseline can be

- a value at time `t_rec` of the baseline time-series `b_data` (if `b = "input"`).
- pre-disturbance values of the state variable in the disturbed system over a period defined by `b_tf` (if `b = "d"`). In that case, the state variable is summarized as the mean or median (`summ_mode`).

For community stability, the distance is calculated as the dissimilarity between the disturbed and baseline communities.

Usage

```
recovery_extent(
  type,
  response = NULL,
  t_rec,
  summ_mode = "mean",
  b = NULL,
  b_tf = NULL,
  vd_i = NULL,
  td_i = NULL,
  d_data = NULL,
  vb_i = NULL,
  tb_i = NULL,
  b_data = NULL,
  comm_d = NULL,
  comm_b = NULL,
  comm_t = NULL,
  method = "bray",
  binary = "FALSE",
  na_rm = TRUE
)
```

Arguments

type	a string defining the type of stability ("functional" or "compositional") to be calculated.
response	a string stating whether the stability metric should be calculated using the log-response ratio between the values in the disturbed system and the baseline (response = "lrr") or using the state variable values in the disturbed system alone.
t_rec	An integer, time point at which the extent of recovery should be calculated.
summ_mode	A string, stating whether the baseline should be summarized as the mean (summ_mode = "mean") or the median (summ_mode = "median"). Defaults to "mean".
b	a string stating whether the baseline is defined by a separate baseline that is specified by the user (b = "input") or by a period of the disturbed system (b = "d") prior to the disturbance. This period is specified by b_tf.
b_tf	a numerical vector, specifying the beginning and end of the pre-disturbance time period for the disturbed time-series that defines the baseline. Obligatory if (b = "d"), see 'Details'.
vd_i	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in d_data.
td_i	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in d_data.
d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in b_data containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in b_data containing said the time variable in the dataframe with baseline values.
b_data	an optional data frame containing the time series of the state variable values in the baseline.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_t	the name of the time variable in comm_b and comm_d.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.

Details

For functional stability, the log-response ratio or difference between the state variable in the disturbed systems v_d and in the baseline (v_b or v_p if the baseline is pre-disturbance values) measured on the user-defined time step when the recovery is assumed to have taken place. Therefore, $R_e = \log\left(\frac{v_d(t)}{v_b(t)}\right)$, or $R_e = \log\left(\frac{v_d(t)}{v_p(t)}\right)$, or $R_e = |v_d(t) - v_b(t)|$, or $R_e = |v_d(t) - v_p(t)|$.

For community stability, the dissimilarity between the disturbed (C_d) and baseline (C_b) communities $R_e = \text{dissim}\left(\frac{C_d(t)}{C_b(t)}\right)$.

Even though it is possible to use a single data value as baseline, it is not recommended, because a single value does not account for any variability in the system arising from, for example, demographic or environmental stochasticity.

Value

A numeric, the extent of recovery. Maximum (functional and compositional) recovery at 0. For functional stability, smaller or higher values indicate under- or overcompensation, respectively. For compositional stability using the Bray-Curtis index (default), $0 \leq R_e \leq 1$. The higher the index, the further apart the communities are after recovery, and thus, the lower the recovery is.

Examples

```
recovery_extent(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps,
  response = "lrr", b = "input", t_rec = 42, vb_i = "statvar_bl",
  tb_i = "time", b_data = aquacomm_resps, type = "functional"
)
recovery_extent(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps,
  response = "diff", b = "input", t_rec = 42, vb_i = "statvar_bl",
  tb_i = "time", b_data = aquacomm_resps, type = "functional"
)
recovery_extent(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps,
  response = "lrr", b = "d", t_rec = 42, b_tf = 8, type = "functional"
)
recovery_extent(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps,
  response = "lrr", b = "d", t_rec = 42, b_tf = c(5, 10), type = "functional"
)
recovery_extent(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps,
  response = "lrr", type = "functional",
  b = "d", t_rec = 42, b_tf = c(5, 10), summ_mode = "median"
)
recovery_extent(
  type = "compositional", t_rec = 28, comm_d = comm_dist,
  comm_b = comm_base, comm_t = "time"
)
```

recovery_rate *Calculate the rate of recovery after disturbance*

Description

recovery_rate (R_r) returns the rate of recovery calculated as the slope of a linear model which uses the time as a predictor of the response. The response can be the state variable in a disturbed system, the log-response ratio or community dissimilarity between the state variable (or community) in the disturbed and baseline systems. The baseline can be

- a value at time t_rec of the baseline time-series b_data (if b = "input").
- pre-disturbance values of the state variable in the disturbed system over a period defined by b_tf (if b = "d"). In that case, the state variable is summarized as the mean or median (summ_mode).

Usage

```
recovery_rate(
  type,
  b = NULL,
  metric_tf,
  response,
  summ_mode = "mean",
  b_tf = NULL,
  vd_i = NULL,
  td_i = NULL,
  d_data = NULL,
  vb_i = NULL,
  tb_i = NULL,
  b_data = NULL,
  comm_d = NULL,
  comm_b = NULL,
  comm_t = NULL,
  method = "bray",
  binary = "FALSE",
  na_rm = TRUE
)
```

Arguments

type	a string defining the type of stability ("functional" or "compositional") to be calculated.
b	a string stating whether the baseline is defined by a separate baseline that is specified by the user (b = "input") or by a period of the disturbed system (b = "d") prior to the disturbance. This period is specified by b_tf.
metric_tf	a numerical vector, specifying the beginning and end of the time period over which the stability metric should be measured.

response	a string stating whether the stability metric should be calculated using the log-response ratio between the values in the disturbed system and the baseline (response = "lrr") or using the state variable values in the disturbed system alone (response == "v").
summ_mode	A string, stating whether the baseline should be summarized as the mean (summ_mode = "mean") or the median (summ_mode = "median"). Defaults to "mean".
b_tf	a numerical vector, specifying the beginning and end of the pre-disturbance time period for the disturbed time-series that defines the baseline. Obligatory if (b = "d"), see 'Details'.
vd_i	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in d_data.
td_i	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in d_data.
d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in b_data containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in b_data containing said the time variable in the dataframe with baseline values.
b_data	an optional data frame containing the time series of the state variable values in the baseline.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_t	the name of the time variable in comm_b and comm_d.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.

Details

For functional stability, the response can the be state variable itself (v_d) , or the log-response ratio between the state variable in the disturbed (v_d) and in the baseline (v_b or v_p if the baseline is pre-disturbance values). For community stability, the response is the dissimilarity between the disturbed (C_d) and baseline (C_b) communities. Therefore,

$$R_r = \frac{\sum(t-\bar{t})(y-\bar{y})}{\sum(t-\bar{t})^2}, \qquad y \in \left\{ v_d,\ \log\left(\frac{v_d}{v_b}\right),\ \log\left(\frac{v_d}{v_p}\right),\ \text{dissim}\left(\frac{C_d}{C_b}\right) \right\}$$

Value

A numeric, the rate of recovery. If $R_r = 0$, the system did not react to the disturbance. If $R_r \geq 0$, the system moved towards the values in the baseline after the disturbance (recovery may be partial). If $Rr \leq 0$, the system deviated even further from the control. In both cases, the higher R_r , the faster the response.

Examples

```
recovery_rate(
  type = "functional", vd_i = "statvar_db", td_i = "time", response = "v",
  d_data = aquacomm_resps, b = "d", metric_tf = c(12, 50)
)
recovery_rate(
  type = "functional", vd_i = "statvar_db", td_i = "time", response = "v",
  d_data = aquacomm_resps, b = "input", metric_tf = c(12, 50),
  vb_i = "statvar_bl", tb_i = "time", b_data = aquacomm_resps
)
recovery_rate(
  type = "compositional", metric_tf = c(0.14, 28), comm_d = comm_dist,
  comm_b = comm_base, comm_t = "time"
)
```

---

resistance *Calculate the resistance of a state variable to disturbance*

Description

`resistance` (R) returns either the distance of a state variable to a baseline value at a specified time point or a maximum distance between the state variables in the disturbed system and the baseline over a specified period. For functional stability, the distance can be calculated as the log-response ratio or as the difference between the state variables in a disturbed time-series and the baseline. For community stability, the distance is calculated as the dissimilarity between the disturbed and baseline communities.

Usage

```
resistance(
  type,
  res_mode = NULL,
  res_time = NULL,
  res_t = NULL,
  res_tf = NULL,
  b = NULL,
  b_tf = NULL,
  vb_i = NULL,
  tb_i = NULL,
  b_data = NULL,
  vd_i = NULL,
```

```
  td_i = NULL,
  d_data = NULL,
  comm_b = NULL,
  comm_d = NULL,
  comm_t = NULL,
  method = "bray",
  binary = "FALSE",
  na_rm = TRUE
)
```

Arguments

type	a string defining the type of stability ("functional" or "compositional") to be calculated.
res_mode	A string stating whether the resistance should be calculated as the log response ratio of the state variable in the disturbed system compared to the baseline (res_mode = "lrr") or the difference (res_mode = "diff") between the values of these state variables. See details.
res_time	A string stating whether resistance should be calculated at a specific point in time (res_time = "defined") or if it should be taken as the maximal difference between the disturbed and baseline state variables over a specified time period (res_time = "max"). Time point and the time period are defined by res_t and res_tf, respectively. See details.
res_t	An integer defining the time point when resistance should be measured if res_time = "defined".
res_tf	A vector, specifying the time period for which the maximum resistance should be looked for, if res_time = "max".
b	a string stating whether the baseline is defined by a separate baseline that is specified by the user (b = "input") or by a period of the disturbed system (b = "d") prior to the disturbance. This period is specified by b_tf.
b_tf	a numerical vector, specifying the beginning and end of the pre-disturbance time period for the disturbed time-series that defines the baseline. Obligatory if (b = "d"), see 'Details'.
vb_i	an optional numeric vector containing the state variable in the baseline, or a string for the name of the column in b_data containing said variable in the dataframe with baseline values.
tb_i	an optional numeric vector containing the time period over which the baseline was measured, or a string for the name of the column in b_data containing said the time variable in the dataframe with baseline values.
b_data	an optional data frame containing the time series of the state variable values in the baseline.
vd_i	a numeric vector containing the state variable in the disturbed system or a string specifying the name of the column containing said variable in the dataframe provided in d_data.
td_i	a numeric vector containing the time or a string specifying the name of the column containing the time in the dataframe provided in d_data.

d_data	an optional data frame containing the time series of the state variable values in a disturbed system.
comm_b	a data frame containing long format community data (species names as columns over time as rows) to calculate compositional metrics.
comm_d	a data frame containing long format community data (species as columns over time as rows) to calculate compositional metrics.
comm_t	an optional string with the name of the time variable in the community data. Only necessary when calculating maximal resistance.
method	a string identifying the dissimilarity index to be used to calculate dissimilarity. For more options, see ?vegdist. Defaults to "bray".
binary	a boolean stating whether presence/absence standardization should be performed before calculating the dissimilarity. For more options, see ?vegdist. Defaults to "bray".
na_rm	a logical determining whether NAs should be taken out prior to the estimation of the stability metric. Defaults to TRUE.

Details

For functional stability, resistance can be calculated as:

- The log response ratio or absolute difference between the state variable's value in the disturbed v_d and the baseline (v_b or v_p if the baseline is pre-disturbance values), on the user-defined time step. Therefore, $R = \log\left(\frac{v_d(t)}{v_b(t)}\right)$, or $R = \log\left(\frac{v_d(t)}{v_p(t)}\right)$, or $R = |v_d(t) - v_b(t)|$, or $R = |v_d(t) - v_p(t)|$.
- The maximal log response ratio or absolute difference between the state variable's value in the disturbed and the baseline systems, over a user-defined time interval: $R = \max_t(\log\left(\frac{v_d(t)}{v_b(t)}\right))$, or $R = \max_t(\log\left(\frac{v_d(t)}{v_p(t)}\right))$, or $R = \max_t(|v_d(t) - v_b(t)|)$, or $R = \max_t(|v_d(t) - v_p(t)|)$.

For compositional stability, the dissimilarity between disturbed (C_d) and baseline (C_b) communities at user-defined time step $R = \text{dissim}\left(\frac{C_d(t)}{C_b(t)}\right)$, or the maximal value $R = \max_t(\text{dissim}\left(\frac{C_d(t)}{C_b(t)}\right))$.

If resistance is calculated at a specific time point, it is conventionally the first time point after the disturbance.

Even though it is possible to use a single data value as baseline (by passing a double to b_tf), it is not recommended, because a single value does not account for any variability in the system arising from, for example, demographic or environmental stochasticity.

Value

A numeric, the resistance value. Maximum (functional and compositional) recovery at 0. For functional stability, smaller or higher values indicate under- or overcompensation, respectively. For compositional stability using the Bray-Curtis index (default), $0 \leq R \leq 1$, and the maximal resistance is 0. The higher the index, the more apart the communities are and thus, the lower resistance is.

Examples

```
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "input",
  vb_i = "statvar_bl", tb_i = "time", b_data = aquacomm_resps,
  res_mode = "lrr", res_time = "defined", res_t = 12, type = "functional"
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "input",
  vb_i = "statvar_bl", tb_i = "time", b_data = aquacomm_resps,
  type = "functional", res_mode = "diff", res_time = "defined", res_t = 12
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "d",
  b_tf = 8, type = "functional", res_mode = "lrr",
  res_time = "defined", res_t = 12
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "d",
  b_tf = 8, type = "functional", res_mode = "diff",
  res_time = "defined", res_t = 12
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "input",
  vb_i = "statvar_bl", tb_i = "time", b_data = aquacomm_resps,
  type = "functional", res_mode = "lrr", res_time = "max", res_tf = c(12, 51)
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "input",
  vb_i = "statvar_bl", tb_i = "time", b_data = aquacomm_resps,
  type = "functional", res_mode = "diff", res_time = "max", res_tf = c(12, 51)
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "d",
  type = "functional", res_mode = "lrr", b_tf = 8, res_time = "max",
  res_tf = c(12, 51)
)
resistance(
  vd_i = "statvar_db", td_i = "time", d_data = aquacomm_resps, b = "d",
  type = "functional", res_mode = "lrr", b_tf = 8, res_time = "max",
  res_tf = c(12, 51)
)
resistance(
  type = "compositional", res_time = "defined", res_t = 28, comm_d = comm_dist,
  comm_b = comm_base, comm_t = "time"
)
```

`stoch_var` *Calculate the intrinsic stochastic invariability of a community from its community matrix.*

Description

stoch_var calculates the intrinsic stochastic (I_S) invariability - a theoretical equivalent of the univariate measure of invariability (Arnoldi et al. 2016).

Usage

```
stoch_var(B)
```

Arguments

B	a matrix, containing the interactions between the species or functional groups in the community. Can be calculated with extractB from the fitted MARSS object.

Details

$$I_S = \frac{1}{\|B^{-1}\|}$$

where $\|B^{-1}\|$ is the spectral norm of the inverse of the matrix B. The spectral norm is computed as the dominant eigenvalue of the matrix

$$B \otimes I + I \otimes B$$

where I is the identity matrix.

Value

A numeric, the stochastic variability. The larger its value, the more stable the system, as its rate of return to equilibrium is higher.

References

Arnoldi, J.-F., Loreau, M., & Haegeman, B. (2016). Resilience, reactivity and variability: A mathematical comparison of ecological stability measures. Journal of Theoretical Biology, 389, 47–59. doi:10.1016/j.jtbi.2015.10.012

Examples

```
library(MARSS)


  # smaller dataset for example:
  # 3 functional groups and two insecticide concentrations besides control
  data_df <- subset(
    aquacomm_fgps,
    treat %in% c(0.0, 0.9, 6) &
      time >= 1 & time <= 28,
    select = c(time, treat, repli, herb, carn, detr)
```

```
)

# estimate z-score transformation and replace zeros with NA
data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                               MARSS::zscore)
data_df[, c("herb", "carn", "detr")] <- lapply(data_df[, c("herb", "carn", "detr")],
                                               function(x) replace(x, x == 0, NA))

# reshape data from wide to long format
data_z_ldf <- reshape(
  data_df,
  varying = list(c("herb", "carn", "detr")),
  v.names = "abund_z",
  timevar = "fgp",
  times = c("herb", "carn", "detr"),
  direction = "long",
  idvar = c("time", "treat", "repli")
)

data_z_ldf <- data_z_ldf[order(data_z_ldf$time, data_z_ldf$treat, data_z_ldf$fgp), ]

# summarize mean and sd
data_z_summldf <- aggregate(abund_z ~ time + treat + fgp, data_z_ldf, function(x)
  c(mean = mean(x, na.rm = TRUE), sd = sd(x, na.rm = TRUE)))
data_z_summldf <- do.call(data.frame, data_z_summldf)
names(data_z_summldf)[4:5] <- c("abundz_mu", "abundz_sd")

# split dataframe per functional groups
# into list to apply the MARSS model more easily
split_data_z <- split(data_z_summldf[, c("time", "fgp", "abundz_mu")], data_z_summldf$treat)

reshape_to_wide <- function(df) {
  df_wide <- reshape(df,
                     idvar = "fgp",
                     timevar = "time",
                     direction = "wide")
  rownames(df_wide) <- df_wide$fgp
  df_wide <- df_wide[, -1]  # Remove the 'fgp' column
  as.matrix(df_wide)
}

data_z_summls <- lapply(split_data_z, reshape_to_wide)

# fit MARSS models
data.marssls <- list(
  MARSS(
    data_z_summls[[1]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
```

```
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[2]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  ),
  MARSS(
    data_z_summls[[3]],
    model = list(
      B = "unconstrained",
      U = "zero",
      A = "zero",
      Z = "identity",
      Q = "diagonal and equal",
      R = matrix(0, 3, 3),
      tinitx = 1
    ),
    method = "BFGS"
  )
)

# identify experiments
names(data.marssls) <- paste0("Conc. = ", c("0", "0.9", "44"), " micro g/L")

# extract community matrices (B)
data.Bls <- data.marssls |>
  lapply(extractB,
         states_names = c("Herbivores", "Carnivores", "Detrivores"))

# calculate intrinsic stochastic variability for each of the B matrices
purrr::map(data.Bls, stoch_var)
```

Index