Package 'coenocliner'

May 8, 2026

Type Package

Title Coenocline Simulation

Version 0.2-4

Maintainer Gavin L. Simpson <ucfagls@gmail.com>

Description Simulate species occurrence and abundances (counts) along gradients.

Imports stats

Suggests testthat, knitr

VignetteBuilder knitr

License GPL-2

URL https://github.com/gavinsimpson/coenocliner/

BugReports https://github.com/gavinsimpson/coenocliner/issues

ByteCompile true

RoxygenNote 7.3.3

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

NeedsCompilation no

Author Gavin L. Simpson [aut, cre],
Jari Oksanen [ctb],
Francisco Rodriguez-Sanchez [ctb]

Repository CRAN

Date/Publication 2025-11-24 12:10:08 UTC

Contents

`coenocline` *Simulate species abundance (counts) or occurrence along one or two gradients*

Description

Simulate species abundance (counts) or occurrence along one or two gradients using well-known ecological response models and random draws from one of a Poisson, negative binomial, Bernoulli, binomial, beta-binomial, zero-inflated Poisson, or zero-inflated negative binomial distribution.

Usage

```
coenocline(
  x,
  responseModel = c("gaussian", "beta"),
  params,
  extraParams = NULL,
 countModel = c("poisson", "negbin", "bernoulli", "binary", "binomial", "betabinomial",
    "ZIP", "ZINB", "ZIB", "ZIBB"),
  countParams = NULL,
  expectation = FALSE
)
```

Arguments

`x`	one of a numeric vector, a list with two components, each a numeric vector, or a matrix with two columns. The vectors are the locations along the gradient(s) at which species responses are to be simulated.
`responseModel`	character; which species response model to use.
`params`	a list of vectors each of which are parameters for the response model for each species. Alternatively, a matrix with one column per parameter and a row for each species.
`extraParams`	a list containing additional parameters required for the response model. Examples include the correlation between gradients in the bivariate Gaussian response model. Components need to be named.

countModel	character; if expectation is FALSE, the default, counts (occurrence) are generated using random deviates from the specified distribution.
countParams	a list of additional parameters required to specify the distribution. An example is the parameter α in the negative binomial distribution. Components need to be named.
expectation	logical; should the expectation (mean) response be returned (TRUE)? If FALSE random counts or occurrences are generated using random draws from a suitably parameterised distribution, as stated in countModel.

Details

coenocline() is a generic interface to coenocline simulation allowing for easy extension and a consistent interface to a range of species response models and statistical distributions.

Two species response models are currently available; the Gaussian response and the generalized beta response model. Random count or occurrence data can be produced via random draws from a suitable distribution; in which case the values obtained from the species response function are used as the expectation of the distribution from which random draws are made.

Parameters for each species in the response model are supplied via argument params and can be provided in one of two ways: i) as a list with named components, each of which is a vector containing values for a single parameter for each species, or ii) as a matrix where each column contains the values for a single parameter and the rows represent species. In each case, the names of the list components or the column names of the matrix must be named for the arguments of the function implementing the species distribution model. See the examples.

Some species response models may require additional parameters not specified at the per species level. An example is the correlation between gradients in the bivariate Gaussian response model. Such parameters are passed via list extraParams and must be named accordingly so that they are passed to the correct argument in the species response function.

The species response model defines the mean of expected response. (In the case of a species occurrence, the probability of occurrence is the expectation.) These represent parameterised distributions. Random count or occurrence data can be produced from these distributions by simulation from those distributions. In this case, a count or probability of occurrence model is used and random draws from the distribution are made. The following distributions are available:

- Poisson,
- Negative binomial,
- Bernoulli,
- Binomial,
- Beta-Binomial,
- Zero-inflated Poisson,
- Zero-inflated Negative binomial,
- Zero-inflated Binomial, and
- Zero-inflated Beta-Binomial

Some distributions may need additional parameters beyond the expectation; an example is the α parameter of (one parameterisation of) the negative binomial distribution. These parameters are species via the list countParams.

Value

a matrix of simulated count or occurrence data, one row per gradient location, one column per species. The object is of class `"coenocline"`, which inherits from the `"matrix"` class.

Additional attributes attached to the matrix are:

`locations` the gradient locations at which response curves were evaluated or for which counts were simulated.

`expectations` the passed value of the `expection`.

`responseModel` the species response model.

`countModel` the count distribution used to simulate counts from.

Author(s)

Gavin L. Simpson

Examples

```
## Poisson counts along a single gradient, Gaussian response
## ==========================================================

x <- seq(from = 4, to = 6, length = 100)
opt <- c(3.75, 4, 4.55, 5, 5.5) + 0.5
tol <- rep(0.25, 5)
h <- rep(20, 5)

## simulate
set.seed(1)
y <- coenocline(x, responseModel = "gaussian",
                params = cbind(opt = opt, tol = tol, h = h),
                countModel = "poisson")
head(y)

y <- coenocline(x, responseModel = "gaussian",
                params = cbind(opt = opt, tol = tol, h = h),
                countModel = "poisson",
                expectation = TRUE)
plot(y, type = "l", lty = "solid")

## Bernoulli distribution (occurrence)
## ===================================

h <- c(1,3,5,7,9) / 10
y <- coenocline(x, responseModel = "gaussian",
                params = cbind(opt = opt, tol = tol, h = h),
                countModel = "bernoulli")
head(y)
## probability of occurrence...
pi <- coenocline(x, responseModel = "gaussian",
                 params = cbind(opt = opt, tol = tol, h = h),
                 countModel = "bernoulli", expectation = TRUE)
```

```
## plot
plot(y, type = "p", pch = 1) # a random realisation
lines(pi, lty = "solid")     # probability of occurrence

## Correlated bivariate Gaussian response, two species
## ==================================================

## gradient locations
x <- seq(3.5, 7, length = 30)
y <- seq(1, 10, length = 30)
xy <- expand.grid(x = x, y = y)

## species parameters on gradients x and y
parx <- list(opt = c(5,6), tol = c(0.5,0.3), h = c(50, 75))
pary <- list(opt = c(5,7), tol = c(1.5, 1.5))

## evaluate response curves at gradient locations
sim <- coenocline(xy, params = list(px = parx, py = pary),
                  responseModel = "gaussian", expectation = TRUE,
                  extraParams = list(corr = 0.5))

## Perspective plots the bivariate responses of the two species
## 'sim' is a matrix 1 column per species with prod(length(x), length(y))
## rows. Need to reshape each species (column) vector into a matrix
## with as many rows as length(x) (number of gradient locations) and
## fill *column*-wise (the default)
persp(x, y, matrix(sim[,1], ncol = length(x)), # spp1
      theta = 45, phi = 30)
persp(x, y, matrix(sim[,2], ncol = length(x)), # spp2
      theta = 45, phi = 30)

## Poisson counts along two correlated gradients, Gaussian response
## ================================================================

set.seed(1)
N <-  100
x1 <- seq(from = 4, to = 6, length = N)
opt1 <- seq(4, 6, length = 5)
tol1 <- rep(0.25, 5)
x2 <- seq(from = 2, to = 20, length = N)
opt2 <- seq(2, 20, length = 5)
tol2 <- rep(1, 5)
h <- rep(30, 5)
xy <- expand.grid(x = x1, y = x2)

set.seed(1)
params <- list(px = list(opt = opt1, tol = tol1, h = h),
               py = list(opt = opt2, tol = tol2))
y <- coenocline(xy,
                responseModel = "gaussian",
                params = params,
                extraParams = list(corr = 0.5),
                countModel = "poisson")
```

```
head(y)
tail(y)

## Visualise one species' bivariate count data
persp(x1, x2, matrix(y[,3], ncol = length(x1)),
      ticktype = "detailed", zlab = "Abundance")

## Recreate beta responses in Fig. 2 of Minchin (1987)
## ==================================================

A0 <- c(5,4,7,5,9,8) * 10
m <- c(25,85,10,60,45,60)
r <- c(3,3,4,4,6,5) * 10
alpha <- c(0.1,1,2,4,1.5,1)
gamma <- c(0.1,1,2,4,0.5,4)
x <- 1:100
params <- list(m = m, A0 = A0, r = r, alpha = alpha, gamma = gamma)

## Expectations
set.seed(2)
y <- coenocline(x, responseModel = "beta",
                params = params,
                countModel = "poisson")
head(y)
plot(y, type = "l", lty = "solid")

y <- coenocline(x, responseModel = "beta",
                params = params,
                countModel = "poisson", expectation = TRUE)
plot(y, type = "l", lty = "solid")

## Zero-inflated Poisson, constant zero-inflation
## ==============================================

y <- coenocline(x, responseModel = "beta", params = params,
                countModel = "ZIP", countParams = list(zprobs = 0.2))
plot(y, type = "l", lty = "solid")

## Zero-inflated Negative binomial, constant zero-inflation
y <- coenocline(x, responseModel = "beta",
                params = params,
                countModel = "ZINB",
                countParams = list(alpha = 0.75, zprobs = 0.2))
plot(y, type = "l", lty = "solid")

## Binomial counts, constant size (m) of 100
## =========================================

## note: A0 must be in range, (0,1)
params["A0"] <- c(5,4,7,5,9,8) / 10
y <- coenocline(x, responseModel = "beta",
                params = params,
```

```
                countModel = "binomial",
                countParams = list(size = 100))
plot(y, type = "l", lty = "solid")

## Beta-Binomial counts, constant size (m) of 100
## ==============================================

## note: A0 must be in range, (0,1)
params["A0"] <- c(5,4,7,5,9,8) / 10
y <- coenocline(x, responseModel = "beta",
                params = params,
                countModel = "betabinomial",
                countParams = list(size = 100, theta = 0.1))
plot(y, type = "l", lty = "solid")
```

---

coenocliner *A coenocline simulation package for R*

Description

coenocliner provides a simple, easy interface for simulating species abundance (counts) or occurrence along gradients.

Details

One of the key ways quantitative ecologists attempt to understand the properties and behaviour of the methods they use or dream up is through the use of simulated data. **coenocliner** is an R package that provides a simple interface to coenocline simulation.

Species data can be simulated from a number of species response models

- Gaussian response
- Generalised Beta response

and random count or occurrence data can be simulated from suitably parameterised response models by using the output from the response model as the mean or expectation of one of a number of statistical distributions

- Poisson
- Negative Binomial
- Bernoulli
- Binomial
- Beta-Binomial
- Zero-inflated Poisson (ZIP)
- Zero-inflated Negative Binomial (ZINB)
- Zero-inflated Binomial (ZIB)
- Zero-inflated Beta-Binomial (ZIBB)

from which random draws are made.

Author(s)

Gavin L. Simpson

See Also

coenocline for simulating species data, distributions for details of the error distributions that can be used for simulations, and species-response for details on the available species response models and the parameters required to use them.

---

distributions *Wrappers to random number generators for use with coenocliner*

Description

These functions are simple wrappers around existing random number generators in R to provide stochastic count data for simulated species.

Usage

```
NegBin(n, mu, alpha)

Poisson(n, mu)

Bernoulli(n, mu)

Binomial(n, mu, size)

BetaBinomial(n, mu, size, theta)

ZIP(n, mu, zprobs)

ZINB(n, mu, alpha, zprobs)

ZIB(n, mu, size, zprobs)

ZIBB(n, mu, size, theta, zprobs)
```

Arguments

n	the number of random draws, equal to number of species times the number of gradient locations.
mu	the mean or expectation of the distribution. For Bernoulli, Binomial, and BetaBinomial() this is the probability of occurrence as given by the response function.

alpha	numeric; dispersion parameter for the negative binomial distribution. May be a vector of length `length(mu)`. The NB2 parametrization of the negative binomial is used here, in which α is positively related to the amount of extra dispersion in the simulated data. As such, where $\alpha = 0$, we would have a Poisson distribution. `alpha` can be supplied a value of `0`, in which case `NegBin` and `ZINB` return random draws from the Poisson or zero-inflated Poisson distributions, respectively. Negative values of `alpha` are not allowed and will generate an error.
size	numeric; binomial denominator, the total number of individuals counted for example
theta	numeric; a positive *inverse* overdispersion parameter for the Beta-Binomial distribution. Low values give high overdispersion. The variance is `size*mu*(1-mu)*(1+(size-1)/(theta+` (Bolker, 2008)
zprobs	numeric; zero-inflation parameter giving the proportion of extraneous zeros. Must be in range $0 \ldots 1$.

Value

a vector of random draws from the stated distribution.

Author(s)

Gavin L. Simpson

References

Bolker, B.M. (2008) *Ecological Models and Data in R.* Princeton University Press.

---

expand

An `expand.grid`*-like function that repeats sets of vectors for every value in a reference vector.*

Description

The values of `x` are repeated for each combination of elements in the vectors supplied via `...`, with the first elements of each vector in `...` being taken as a set, the second elements as another set, and so on. `x` is repeated for each of these sets.

Usage

```
expand(x, ...)
```

Arguments

x	numeric; vector of data points which are to be replicated for each of the sets of vectors supplied to `...`.
...	additional vector arguments to be expanded to the correct length. These are taken to be a set of values to be replicated for each of the elements of `x`.

Value

a matrix of replicated vectors, with column names for `x` and named arguments passed as `...`.

Author(s)

Gavin L. Simpson

References

Minchin P.R. (1987) Simulation of multidimensional community patterns: towards a comprehensive model. *Vegetatio* **71**, 145–156.

Examples

```
# Recreate Fig. 2 of Minchin (1987)
# Parameters for each of 6 six species
A0 <- c(5,4,7,5,9,8) * 10
m <- c(25,85,10,60,45,60)
r <- c(3,3,4,4,6,5) * 10
alpha <- c(0.1,1,2,4,1.5,1)
gamma <- c(0.1,1,2,4,0.5,4)
# Gradient locations
x <- 1:100

# expand parameter set
pars <- expand(x, m = m, A0 = A0, r = r, alpha = alpha,
               gamma = gamma)
head(pars)
```

`locations` *Extract Gradient Locations*

Description

Extract the gradient locations at which response curves were evaluated or for which counts were simulated.

Usage

```
locations(x, ...)

## Default S3 method:
locations(x, ...)
```

Arguments

`x`	an object with `locations` as an attribute or a component, such as the object returned by `coenocline`.
`...`	arguments passed to other methods.

Value

A vector or a matrix of gradient locations. For single-gradient simulations, a vector is returned, whereas for two-gradient simulations, a matrix of location pairs is returned.

Author(s)

Gavin L. Simpson

Examples

```
## Poisson counts along a single gradient, Gaussian response
## ==========================================================

x <- seq(from = 4, to = 6, length = 100)
opt <- c(3.75, 4, 4.55, 5, 5.5) + 0.5
tol <- rep(0.25, 5)
h <- rep(20, 5)

## simulate
set.seed(1)
y <- coenocline(x, responseModel = "gaussian",
                params = cbind(opt = opt, tol = tol, h = h),
                countModel = "poisson")
head(locations(y))
```

---

persp.coenocline *Perspective Plot of Species Simulations Along Gradients*

Description

A simple S3 persp method for coenocline simulations.

Usage

```
## S3 method for class 'coenocline'
persp(x, species = NULL, theta = 45, phi = 30, ...)
```

Arguments

x	an object of class "coenocline", the result of a call to coenocline.
species	vector indicating which species to plot. This can be any vector that you can use to subset a matrix, but numeric or logical vectors would be mostly commonly used.
theta, phi	angles defining the viewing direction. theta gives the azimuthal direction and phi the colatitude. See persp for further details.
...	additional arguments to persp.

Value

A plot is drawn on the current device.

Author(s)

Gavin L. Simpson

Examples

```
## Poisson counts along two correlated gradients, Gaussian response
## ================================================================

set.seed(1)
N <-  40
x1 <- seq(from = 4, to = 6, length = N)
opt1 <- seq(4, 6, length = 5)
tol1 <- rep(0.25, 5)
x2 <- seq(from = 2, to = 20, length = N)
opt2 <- seq(2, 20, length = 5)
tol2 <- rep(1, 5)
h <- rep(30, 5)
xy <- expand.grid(x = x1, y = x2)

set.seed(1)
params <- list(px = list(opt = opt1, tol = tol1, h = h),
               py = list(opt = opt2, tol = tol2))
y <- coenocline(xy,
                responseModel = "gaussian",
                params = params,
                extraParams = list(corr = 0.5),
                countModel = "poisson")

## perspective plot(s) of simulated counts
layout(matrix(1:6, ncol = 3))
op <- par(mar = rep(1, 4))
persp(y)
par(op)
layout(1)

## as before but now just expectations
y <- coenocline(xy,
                responseModel = "gaussian",
                params = params,
                extraParams = list(corr = 0.5),
                countModel = "poisson",
                expectation = TRUE)

## perspective plots of response curves
layout(matrix(1:6, ncol = 3))
op <- par(mar = rep(1, 4))
persp(y)
par(op)
```

```
layout(1)

## Same plots generated using the `plot` method
layout(matrix(1:6, ncol = 3))
op <- par(mar = rep(1, 4))
persp(y)
par(op)
layout(1)
```

`plot.coenocline` *Plot species simulations along gradients*

Description

A simple S3 `plot` method for coenocline simulations.

Usage

```
## S3 method for class 'coenocline'
plot(x, type = "p", pch = 1, ...)

## S3 method for class 'coenocline'
lines(x, lty = "solid", ...)
```

Arguments

`x`	an object of class `"coenocline"`, the result of a call to `coenocline`.
`type`	character; the type of plot to produce. See `plot.default` for details.
`pch`	the plotting character to use. See `plot.default` for details.
`...`	additional arguments to `matplot`.
`lty`	the line type to use. See `plot.default` for details.

Value

A plot is drawn on the current device.

Author(s)

Gavin L. Simpson

Examples

```
## Poisson counts along a single gradient, Gaussian response
## ==========================================================

x <- seq(from = 4, to = 6, length = 100)
opt <- c(3.75, 4, 4.55, 5, 5.5) + 0.5
tol <- rep(0.25, 5)
```

```
h <- rep(20, 5)

## simulate
set.seed(1)
y <- coenocline(x, responseModel = "gaussian",
                params = cbind(opt = opt, tol = tol, h = h),
                countModel = "poisson")
head(y)

y <- coenocline(x, responseModel = "gaussian",
                params = cbind(opt = opt, tol = tol, h = h),
                countModel = "poisson",
                expectation = TRUE)
plot(y, type = "l", lty = "solid")
```

---

showParams *List parameters of species response models*

Description

Returns the parameters of the indicated response model.

Usage

```
showParams(model = c("gaussian", "beta"))
```

Arguments

model	character; the species response model for which parameters will be listed

Value

A character vector of parameters. The species response model is returned as attribute `"model"`. Attribute `"onlyx"` is a logical vector indicating which, if any, of the parameters are intended to be supplied only once per species and not for both gradients.

Author(s)

Gavin L. Simpson

See Also

`Gaussian` and `Beta` for the species response model functions themselves.

Examples

```
showParams("gaussian")
```

simJamil *Simulate species abundance data following Jamil & ter Braak (2013)*

Description

Simulate species probability of occurrence data according to the method used by Tahira Jamil and Cajo ter Braak in their recent paper *Generalized linear mixed models can detect unimodal species-environment relationships*.

Usage

```
simJamil(
  n,
  m,
  x,
  gl = 4,
  randx = TRUE,
  tol = 0.5,
  tau = gl/2,
  randm = TRUE,
  expectation = FALSE
)
```

Arguments

n	numeric; the number of samples/sites.
m	numeric, the number of species/variables.
x	numeric; values for the environmental gradient. Can be missing, in which case suitable values are generated. See Details.
gl	numeric; gradient length in arbitrary units. The default is 4 units with gradient values ranging from -2 to 2.
randx	logical; should locations along the gradient (x) be located randomly or equally-spaced?
tol	numeric; the species tolerances. Can be a vector of length m, hence allowing for varying tolerances along the gradient x.
tau	numeric; constant that ensures some of the optima are located beyond the observed gradient end points.
randm	logical; should species optima along the gradient be located randomly or equally-spaced?
expectation	logical; if TRUE the binomial probabilities p_{ij} from the response curve are returned directly. If FALSE, the default, random draws from a Bernoulli distribution with probability p_{ij} are made.

Value

a matrix of n rows and m columns containing the simulated species abundance data.

Author(s)

Gavin L. Simpson

References

Jamil and ter Braak (2013) Generalized linear mixed models can detect unimodal species-environment relationships. *PeerJ* **1:e95**; DOI doi:10.7717/peerj.95.

Examples

```
set.seed(42)
N <- 100   # Number of locations on gradient (samples)
glen <- 4  # Gradient length
grad <- sort(runif(N, -glen/2, glen/2)) # sample locations
M <- 10    # Number of species
sim <- simJamil(n = N, m = M, x = grad, gl = glen, randx = FALSE,
                randm = FALSE, expectation = TRUE)
## visualise the response curves
matplot(grad, sim, type = "l", lty = "solid")

## simulate binomial responses from those response curves
sim <- simJamil(n = N, m = M, x = grad, gl = glen, randx = FALSE,
                randm = FALSE)
```

`species-response` *Species response models for coenocline simulation*

Description

Parameterise species response curves along one or two gradients according to a Gaussian or generalised beta response model.

Usage

```
Gaussian(x, y = NULL, px, py = NULL, corr = 0)

Beta(x, y = NULL, px, py = NULL)
```

Arguments

`x`	numeric; locations of observations on the primary gradient.
`y`	numeric; locations of observations on the secondary gradient. Can be missing is only a single gradient is required.
`px`	a list of named elements, each of which is a vector of numeric parameter values for the species response on the primary gradient `x`. See Details for further information on the required parameters.

py	a list of named elements, each of which is a vector of numeric parameter values for the species response on the secondary gradient y. See Details for further information on the required parameters.
corr	numeric; the correlation between gradients x and y. Only applies to Gaussian().

Details

Gaussian() and Beta() return values from appropriately parameterised Gaussian or generalised beta response models respectively. Parameters for the primary (x) and secondary (y) gradients are supplied as lists via arguments px and py. Parameters are supplied in the form of vectors, one per parameter. These vectors must be supplied to named components in the respective lists. The names of the components must match the parameters of the required response model.

For Gaussian() the following named components must be supplied:

opt the species optima

tol the species tolerances

h the heights of the response curves at the optima. This parameter should only be supplied to px; in the case of simulations along two gradients, the height of the response curve applies to both gradients and is the height of a bivariate Gaussian distribution at the bivariate optima.

For Beta() the following named components must be supplied:

A0 The heights of the species response curves at their modes. Like the parameter h for the Gaussian response, this parameter should only be passed via px; in the case of simulations along two gradients, the height of the response curve applies to both gradients and is the height of a bivariate generalised beta distribution at the bivariate mode.

m the locations on the gradient of the modal abundance (the species optima)

r the ranges of occurrence of species on the gradient

alpha a shape parameter. With gamma, alpha informs the shape of the response curve and control the skewness and kurtosis of the curve. Only positive values are allowed, which lead to unimodal response curves. If alpha is equal to gamma, the species response curve is symmetric, otherwise an asymmetric curve is generated.

gamma a shape parameter. With alpha, gamma informs the shape of the response curve and control the skewness and kurtosis of the curve. Only positive values are allowed, which lead to unimodal response curves. If gamma is equal to alpha, the species response curve is symmetric, otherwise an asymmetric curve is generated.

See the examples here and in coenocline for details on how to set up calls to these species response functions.

Value

A numeric vector of species "abundances" of length equal to length(x).

Author(s)

Gavin L. Simpson

Examples

```
# A simple example with a single species
x <- seq(from = 4, to = 6, length = 100)
px <- list(opt = 4.5, tol = 0.25, h = 20)
G <- Gaussian(x, px = px)
head(G)
length(G)

# A more complex example with 6 species, which needs the parameters
# repeating for each gradient location:

# Recreate Fig. 2 of Minchin (1987)
# Parameters for each of 6 six species
A0 <- c(5,4,7,5,9,8) * 10
m <- c(25,85,10,60,45,60)
r <- c(3,3,4,4,6,5) * 10
alpha <- c(0.1,1,2,4,1.5,1)
gamma <- c(0.1,1,2,4,0.5,4)
# Gradient locations
x <- 1:100

# expand parameter set
pars <- expand(x, m = m, A0 = A0, r = r, alpha = alpha,
gamma = gamma)
head(pars)

xvec <- pars[, "x"]
px <- as.list(data.frame(pars[, -1]))
spprc <- Beta(xvec, px = px)
matplot(matrix(spprc, ncol = 6), ## 6 species
        type = "l", lty = "solid")

# Bivariate beta, single species
xx <- 1:100
yy <- 1:100
xy <- expand.grid(x = xx, y = yy)
parx <- expand(xy[, "x"], A0 = 50, m = 60, r = 40, alpha = 4, gamma = 4)
pary <- expand(xy[, "y"], m = 60, r = 40, alpha = 4, gamma = 4)

x <- parx[,1]
px <- as.list(as.list(data.frame(parx[, -1])))
y <- pary[,1]
py <- as.list(as.list(data.frame(pary[, -1])))

spprc <- Beta(x, y, px = px, py = py)
persp(xx, yy, matrix(spprc, ncol = length(xx)))
```

`stack.coenocline` *Stack columns of a species coenocline simulation*

Description

Stacks columns of a species coenocline simulation into long format suitable for use in statistical modelling or ggplot/lattice plots.

Usage

```
## S3 method for class 'coenocline'
stack(x, ...)
```

Arguments

x	an object of class "coenocline"
...	arguments passed to other methods (not used).

See Also

coenocline

Index