Package: ecolottery (via r-universe)

September 26, 2024

Type Package

Title Coalescent-Based Simulation of Ecological Communities

Version 2.0.0

URL https://github.com/frmunoz/ecolottery

BugReports https://github.com/frmunoz/ecolottery/issues

BuildVignettes true

VignetteBuilder knitr

Depends R (>= 3.0.2)

Imports abc, ade4, data.table, plyr, stats, graphics, ggplot2, grDevices, hillR, parallel, EasyABC, lazyeval, moments, pbapply, Rcpp

Suggests ape, dplyr, e1071, changepoint, knitr, picante, RColorBrewer, reshape, rmarkdown, sads, testthat, utils, vegan, Weighted.Desc.Stat

#LinkingTo Rcpp

Description Coalescent-Based Simulation of Ecological Communities as proposed by Munoz et al. (2018) <doi:10.1111/2041-210X.12918>. The package includes a tool for estimating parameters of community assembly by using Approximate Bayesian Computation.

License GPL-3

Encoding UTF-8

RoxygenNote 6.1.1

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

RemoteUrl https://github.com/frmunoz/ecolottery

RemoteRef HEAD

RemoteSha cbd4fe296d00768fe8a523430a26a359e7d41fea

Contents

`ecolottery-package` *Coalescent-Based Simulation of Ecological Communities*

Description

Coalescent-Based Simulation of Ecological Communities as proposed by Munoz et al. (2018) <doi:10.1111/2041-210X.12918>. The package includes a tool for estimating parameters of community assembly by using Approximate Bayesian Computation.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.
Two basic functions: `coalesc` for coalescent-based simulation, and `forward` for forward-in-time simulation

Author(s)

François Munoz [aut, cre], Matthias Grenié [aut], Pierre Denelle [aut], Elizabeth Barthelemy [aut], Adrien Taudière [ctb], Fabien Laroche [ctb], Caroline Tucker [ctb], Cyrille Violle [ctb]

Maintainer: François Munoz <francois.munoz@hotmail.fr>

References

Hurtt, G. C. and S. W. Pacala (1995). "The consequences of recruitment limitation: reconciling chance, history and competitive differences between plants." Journal of Theoretical Biology 176(1): 1-12.

Hubbell, S. P. (2001). "The Unified Neutral Theory of Biodiversity". Princeton University Press.

Gravel, D., C. D. Canham, M. Beaudet and C. Messier (2006). "Reconciling niche and neutrality: the continuum hypothesis." Ecology Letters 9(4): 399-409.

Munoz, F., P. Couteron, B. R. Ramesh and R. S. Etienne (2007). "Estimating parameters of neutral communities: from one Single Large to Several Small samples." Ecology 88(10): 2482-2488.

Munoz, F., B. R. Ramesh and P. Couteron (2014). "How do habitat filtering and niche conservatism affect community composition at different taxonomic resolutions?" Ecology 95(8): 2179-2191.

Examples

```
## Coalescent-based simulation of stabilizing habitat filtering around
## t = 0.5
J <- 100; theta <- 50; m <- 0.5;
comm <- coalesc(J, m, theta, filt = function(x) 0.5 - abs(0.5 - x))
plot_comm(comm)

## Forward-in-time simulation of stabilizing habitat filtering around
## t = 0.5, over 100 time steps

# A regional pool including 100 species each including 10 individuals
pool <- sort(rep(as.character(1:100), 10))

# Initial community composed of 10 species each including 10 individuals,
# with trait information for niche-based dynamics
initial <- data.frame(ind = paste("init", 1:100, sep="."),
                      sp = sort(rep(as.character(1:10), 10)),
                      trait = runif(100))
final <- forward(initial = initial, m = 0.5, gens = 100, pool = pool,
                 filt = function(x) 0.5 - abs(0.5 - x))
plot_comm(final)
```

---

abund *Compute absolute and relative abundances in the local community and the reference pool*

Description

Compute the abundances and relative abundances of species in simulated communities and in the corresponding species pools. The input must be an output of either `coalesc` or the `forward` functions.

Usage

```
abund(x)
```

Arguments

x	a list including the species pool composition (`x$pool`) and the local community composition (`x$com`)

Value

pool	species abundances and relative abundances in the reference pool
com	species abundances and relative abundances in the local community

Author(s)

F. Munoz, P. Denelle and M. Grenie

Examples

```
# Simulation of a neutral community including 500 individuals
J <- 500; theta <- 50; m <- 0.05;
comm1a <- coalesc(J, m, theta)
abund1a <- abund(comm1a)

# Log-series distribution of regional abundances
fit <- vegan::fisherfit(abund1a$pool$ab)
freq <- as.numeric(names(fit$fisher))
plot(log(freq), fit$fisher,
     xlab = "Frequency (log)",
     ylab = "Species", type = "n")
rect(log(freq - 0.5), 0, log(freq + 0.5), fit$fisher, col="skyblue")

alpha <- fit$estimate
k <- fit$nuisance

curve(alpha * k^exp(x) / exp(x), log(0.5), max(log(freq)),
      col = "red", lwd = 2, add = TRUE)

# Relationship between local and regional abundances
par(mfrow=c(1, 2))
plot(abund1a$pool[rownames(abund1a$com), "relab"],
  abund1a$com$relab,
  main = "m = 0.05",
  xlab = "Regional abundance",
  ylab = "Local abundance",
  log = "xy")
abline(0,1)

# With higher immigration rate
m <- 0.95
comm1b <- coalesc(J, m, theta)
abund1b <- abund(comm1b)
plot(abund1b$pool[rownames(abund1b$com),"relab"],
  abund1b$com$relab,
  main = "m = 0.95",
  xlab = "Regional abundance",
  ylab = "Local abundance",
  log = "xy")
abline(0,1)
```

coalesc *Coalescent-based simulation of ecological communities undergoing both neutral and niche-base dynamics*

Description

Simulates the composition of a community based on immigration from a regional pool, habitat filtering depending on local environment and species traits, and local birth-death stochastic dynamics.

Usage

```
coalesc(J, m = 1, theta = NULL, filt = NULL, filt.vect = F, m.replace = T,
        add = FALSE, var.add =NULL, pool = NULL, traits = NULL,
        Jpool = 50 * J, verbose = FALSE, checks = TRUE)
```

Arguments

J	number of individuals in the local community.
m	migration rate (if m = 1 the community is a subsample of the regional pool).
theta	parameter of neutral dynamics in the regional pool (used only if pool=NULL), it is the "fundamental biodiversity number" (θ).
m.replace	should the immigrants be drawn with replacement from the source pool. Default is TRUE.
filt	a function representing the effect of local habitat filtering. For an individual that displays the value(s)t of a single or several trait(s), filt(t) represents the probability that the individual enters the local community. If filt = NULL, coalesc() provides a neutral community.
filt.vect	indicates whether the filtering function can be vectorized. It means that the function can take as input a vector of trait values and provide a vector of the corresponding weights.
add	indicates if additional variables must be passed to filt. It can be, for instance, environmental data conditioning the trait-based filtering in the community. Default is FALSE.
var.add	additional variables to be passed to filt when add = T.
pool	the regional pool of species providing immigrants to the local community. It should include the label of individual on first column, and of its species on second column. If pool = NULL, the pool is simulated as a metacommunity at speciation-drift equilibrium, based on parameter theta. The provided pool can contain trait values for each individuals in a third column.
traits	a matrix or data.frame including one or several traits on columns. A unique trait value is assigned to each species in the regional pool. Species names of pool must be included in traits. If traits = NULL and trait information is absent from pool, a random trait value is given to species of the regional pool, from a uniform distribution between 0 and 1.

Jpool	if pool = NULL, it is the number of individuals to be simulated in the regional pool.
verbose	if verbose = TRUE, functions returns a lot of outputs about parameters, species pool and environmental filter.
checks	should initial checks that the inputs are correct be performed?

Details

Coalescent-based simulation of a community of size J. This generic function can simulate a neutral community (if filt = NULL) or a community undergoing both neutral and niche-based dynamics. In the latter case, filt(t) represents the relative ability of immigrants with trait values t in the regional pool to enter the community.

If trait-based filtering involves several traits, filt(t) must be defined such as t is a vector of the trait values of a candidate immigrant. See example below.

A Shiny app is available to visualize simulated trait distributions for chosen parameter values in the model.

Value

com	a data.frame of simulated individuals, with the label of ancestor individual in the regional pool on first column (as in first column of pool), species label on second column (as in second column of pool), and species trait (as in third column of pool). **Not provided if** m = 1 **and** filt = NULL**: in this case the function provides a sample of the regional pool.**
pool	a data.frame of the individuals of the regional source pool, with the label of ancestor individual in the regional pool on first column (as in first column of input pool), species label on second column (as in second column of input pool), and species trait (as in third column of input pool).
call	the call function.

Author(s)

F. Munoz

References

Hurtt, G. C. and S. W. Pacala (1995). "The consequences of recruitment limitation: reconciling chance, history and competitive differences between plants." Journal of Theoretical Biology 176(1): 1-12.

Gravel, D., C. D. Canham, M. Beaudet and C. Messier (2006). "Reconciling niche and neutrality: the continuum hypothesis." Ecology Letters 9(4): 399-409.

Munoz, F., P. Couteron, B. R. Ramesh and R. S. Etienne (2007). "Estimating parameters of neutral communities: from one Single Large to Several Small samples." Ecology 88(10): 2482-2488.

Munoz, F., B. R. Ramesh and P. Couteron (2014). "How do habitat filtering and niche conservatism affect community composition at different taxonomic resolutions?" Ecology 95(8): 2179-2191.

Examples

```
# Simulation of a neutral community including 100 individuals
J <- 500; theta <- 50; m <- 0.1
comm1 <- coalesc(J, m, theta)
# Regional and local trait distributions
plot_comm(comm1)

# Define a regional pool of species with equal abundances
pool <- cbind(1:10000, rep(1:500, 20), rep(NA, 10000))
# Uniform distribution of trait values
t.sp <- runif(500)
# No intraspecific variation
pool[,3] <- t.sp[pool[,2]]
# Generate a neutral community drawn from the pool
comm2<- coalesc(J, m, pool = pool)
plot_comm(comm2)

# Directional habitat filtering toward t = 0
comm3 <- coalesc(J, m, filt = function(x) 1 - x, pool = pool)
# Regional and local trait distributions
plot_comm(comm3)

# Function for environmental filtering
sigma <- 0.1
filt_gaussian <- function(t, x) exp(-(x - t)^2/(2*sigma^2))

# Stabilizing habitat filtering around t = 0.1
comm4a <- coalesc(J, m, filt = function(x) filt_gaussian(0.1, x), pool = pool)
plot_comm(comm4a)
# Stabilizing habitat filtering around t = 0.5
comm4b <- coalesc(J, m, filt = function(x) filt_gaussian(0.5, x),
                  pool = pool)
plot_comm(comm4b)
# Stabilizing habitat filtering around t = 0.9
comm4c <- coalesc(J, m, filt = function(x) filt_gaussian(0.9, x),
                  pool = pool)
plot_comm(comm4c)

# Mean trait values in communities reflect the influence of habitat filtering
mean(comm4a$com[, 3])
mean(comm4b$com[, 3])
mean(comm4c$com[, 3])

# Disruptive habitat filtering around t = 0.5
comm5 <- coalesc(J, m, filt = function(x) abs(0.5 - x), pool = pool)
plot_comm(comm5)

# Multi-modal habitat filtering
t.sp <- rnorm(500)
pool[, 3] <- t.sp[pool[,2]]
comm6 <- coalesc(J, m, filt = function(x) sin(3*x) + 1, pool = pool)
plot_comm(comm6)
```

```
# Filtering depending on multiple traits
# We define two traits
t1.sp <- runif(500)
t2.sp <- runif(500)
pool[, 3] <- t1.sp[pool[,2]]
pool <- cbind(pool, t2.sp[pool[,2]])
# Here the probability of successful immigration depends
# on the product of Gaussian filters playing on two traits,
# with different optimal values
comm7 <- coalesc(J, m, filt = function(x)
filt_gaussian(0.75, x[1])*filt_gaussian(0.25, x[2]),
pool = pool)
# Distribution of trait 1
plot_comm(comm7, seltrait = 1)
# Distribution of trait 2
plot_comm(comm7, seltrait = 2)
```

`coalesc_abc` *Estimation of parameters of community assembly using Approximate Bayesian Computation (ABC)*

Description

Estimates parameters of neutral migration-drift dynamics (through migration rate m and parameters of environmental filtering (through a filtering function `filt.abc()`) from the composition of a local community and the related regional pool.

Usage

```
coalesc_abc(comm.obs, pool = NULL, multi = "single", prop = FALSE, traits = NULL,
f.sumstats, filt.abc = NULL,  filt.vect = F, migr.abc = NULL, m.replace = TRUE,
size.abc = NULL, add = FALSE, var.add = NULL, params = NULL, par.filt = NULL,
par.migr = NULL, par.size = NULL, constr = NULL, scale = FALSE, dim.pca = NULL,
svd = FALSE, theta.max = NULL, nb.samp = 10^6, parallel = FALSE, nb.core = NULL,
tol = NULL, type = "standard", method.seq = "Lenormand",
method.mcmc = "Marjoram_original", method.abc = "rejection", alpha = 0.5,
pkg = NULL)

initial_checks(comm.obs = NULL, pool = NULL, multi = "single", prop = F, traits = NULL,
f.sumstats = NULL, filt.abc = NULL, filt.vect = F, migr.abc = NULL, size.abc = NULL,
add = NULL, var.add = NULL, params = NULL,  par.filt = NULL, par.migr = NULL,
par.size = NULL, constr = NULL, scale = F, dim.pca = NULL, svd = NULL,
theta.max = NULL, nb.samp = 10^6, parallel = F, nb.core = NULL, tol = NULL,
type = "standard", method.seq = "Lenormand", method.mcmc = "Marjoram_original",
method.abc = "rejection", alpha = 0.5, pkg = NULL)
```

Arguments

comm.obs	the observed community composition. If multi = "single" (default), should be a matrix or data.frame of individuals on rows with their individual id (first column), and species id (second column).
pool	composition of the regional pool to which the local community is hypothesized to be related through migration dynamics with possible environmental filtering. Should be a matrix of individuals on rows with their individual id (first column), species id (second column), and (optionally) the trait values of the individuals. When multi = "tab" or "seqcom", different pools can be set for each community: pool must be then defined as a list of matrices with the same columns as above.
multi	structure of the community inputs: • if multi = "single", comm.obs represents a single community • if multi = "tab", the user provides a site-species matrix (**sites in rows and species in columns**) • if multi = "seqcom", comm.obs contains a list of communities. If prop = F, the local community composition is given with one individual per row (first column is individual id, second column is species id, subsequent columns provide trait values). If prop = F, the local community composition is given with one species per row (first column if species id, second column is coverage, subsequent columns provide trait values).
prop	indicates if the community composition is given in term of relative species abundances, cover or proportions. In this case, a parameter of effective community size is estimated. Default is FALSE. If no prior is provided for J based on argumeng par.size, it is by default a uniform distribution between 100 and 1000.
traits	the trait values of species in the regional pool. It is used if trait information is not provided in pool. In this case, intraspecific trait variation is assumed to be null. Species names of pool must be included in traits.
f.sumstats	a function calculating the summary statistics of community composition. It is be used to compare observed and simulated community composition in ABC estimation. The first input argument of the function concerns community composition, under a format depending on multi and on prop. If multi="comm" this first argument is the local community composition with individual and species labels as first columns and (optionally) trait values as subsequent columns. There can be a second argument taking the average trait values per species (derived from pool if not provided in traits, see above), and var.add (see below) can be sent as a third argument. See examples of different options below. The function should return a vector of summary statistics.
filt.abc	the hypothesized environmental filtering function. It is a function of individual trait values (first argument), additional parameters to be estimated and (optionally) environmental variables defined in var.add. If NULL, neutral communities will be simulated and only m will be estimated.
filt.vect	indicates whether the filtering function can be vectorized. It means that the function can take as input a vector of trait values and provide a vector of the corresponding weights.

migr.abc — a function defining immigration probability in local communities. It can be a function of environmental variables defined in var.add and of parameters to be estimated.

m.replace — should the immigrants be drawn with replacement from the source pool. Default is TRUE.

size.abc — a function defining local community sizes. It can be used when prop = T. It can be a function of environmental variables defined in var.add and of parameters to be estimated.

add — indicates if additional variables must be passed to filt.abc and/or migr.abc . It can be, for instance, environmental data conditioning the trait-based filtering and/or immigration in the community. Default is FALSE.

var.add — additional variables to be passed to filt.abc and/or migr.abc when add = T.

params — equivalent to par.filt (see below): it is kept in the function for compatibility with previous versions.

par.filt — a matrix of the bounds of the parameters used in filt.abc. The row names of par.filt provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

par.migr — a matrix of the bounds of the parameters used in migr.abc. The row names of par.migr provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

par.size — a matrix of the bounds of the parameters used in size.abc. The row names of par.size provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

constr — constraints that must be set on parameter values, i.e., relationships that must be met when drawing the values in prior distributions. It must be a vector of character strings each including an operator relating some of the parameters. The names of parameters must be consistent with those used in par.filt, par.migr and par.size. Default is NULL (no constraint).

scale — should the summary statistics be scaled. Default is FALSE.

dim.pca — gives a number of independent dimensions to calculate from the simulated summary statistics, by performing Principal Component Analysis (PCA). Default is NULL, in which case no PCA is performed. If svd = T, Singular Value Decomposition (SVD) will be performed instead of PCA.

svd — indicates if Singular Value Decomposition (SVD) must be performed to get the basic dimensions requested in dim.pca. Will be ignored if dim.pca = NULL.

theta.max — if pool = NULL, regional abundances will be simulated following a log-series distribution. The function will estimate the theta parameter of this distribution. theta.max then provides the upper bound for this estimation.

nb.samp — the number of parameter values to be sampled in ABC calculation. Random values of parameters of environmental filtering (see filt.abc and params) and of migration (denoted as m) are drawn from a uniform distribution between minimum and maximum values provided in params (and between 0 and 1 for m).

parallel	boolean. If parallel = TRUE, the function will perform parallel processing using the parLapply() function of package parallel. Default is parallel = FALSE.
nb.core	number of cores to be used in parallel computation if parallel = TRUE. If NULL (default), all the cores minus 1 are used. If 1, only one core is used (i.e., no parallel computing).
tol	the tolerance value used in ABC estimation (see help in abc() function of package abc for further information). Default is NULL.
pkg	packages needed for calculation of filt.abc and/or f.sumstats.
type	the type of algorithm to be used in EasyABC. Can be either "standard" (using package abc, default)" "seq" (sequential), "mcmc" or "annealing". Three later options are based on the EasyABC package.
method.seq	when type = "seq", gives the algorithm for sequential sampling scheme, which is passed to ABC_sequential. Can be "Lenormand" (Default), "Drovandi", "Delmoral", "Beaumont" or "Emulation".
method.mcmc	when type = "mcmc", gives the algorithm for MCMC sampling scheme, which is passed to ABC_mcmc. Can be "Marjoram_original"(Default), "Marjoram" or "Wegmann". The method "Marjoram_original" cannot be used with multiple cores.
method.abc	the method to be used in ABC estimation (see help on abc() function of package abc for further information). Can be "rejection", "loclinear", "neuralnet" or "ridge". Default is "rejection".
alpha	a positive number between 0 and 1 (strictly) used when performing sequential ABC method. alpha is the proportion of particles rejected at each step in the algorithm "Drovandi". This is the proportion of particles kept at each step in the algorithms "Delmoral", "Lenormand" and "Emulation". Default value is 0.5

Details

coalesc_abc() performs ABC estimation for one (if multi = "single", default) or several communities (if multi = "tab" or "seqcom") related to the same regional pool. If type = "standard", it performs simulations with parameter values randomly drawn in priori distributions. With other options in type, the function uses optimization algorithms implemented in the package EasyABC.

To assess environmental filtering, a filtering function must be defined in filt.abc. It should take two arguments, the first is a trait value of a candidate immigrant, the second is a vector including the parameter values of the filtering function. See examples below for further information on usage. An important point is that in many cases the function might be vectorized, that is, it can provide a vector of filtering probabilities for a vector of trait values given in first argument. In this case the user should set filt.vect = T, which will significantly accelerate simulations.

The related functions coalesc_abc_std() and coalesc_abc_std() perform simulations with the option type = "standard" and with the algorithms from EasyABC package, respectivelt.

initial_checks() performs initial checks of the input arguments of coalesc_abc. It provides informative error and warning messages when needed. This function is not intended to be used directly.

Value

par	parameter values used in simulations.
obs	observed summary statistics.
obs.scaled	observed summary statistics standardized according to the mean and standard deviation of simulated values.
ss	standardized summary statistics of the communities simulated with parameter values listed in par.
ss.scale	data frame including the mean and the standard deviation used for standardization of observed and summary statistics.
abc	a single (if multi = FALSE, default) or a list of abc objects including ABC estimation information for each community provided in input (comm.obs). It is of class abc for type="standard" and of class EasyABC otherwise.

Author(s)

F. Munoz, E. Barthelemy

References

Jabot, F., and J. Chave. 2009. Inferring the parameters of the neutral theory of biodiversity using phylogenetic information and implications for tropical forests. Ecology Letters 12:239-248.

Csillery, K., M. G. B. Blum, O. E. Gaggiotti, and O. Francois. 2010. Approximate Bayesian computation (ABC) in practice. Trends in Ecology & Evolution 25:410-418.

Csillery, K., O. Francois, and M. G. Blum. 2012. abc: an R package for Approximate Bayesian Computation (ABC). Methods in Ecology and Evolution 3:475-479.

Jabot, F., T. Faure, and N. Dumoulin 2013. EasyABC: performing efficient approximate Bayesian computation sampling schemes using R. Methods in Ecology and Evolution 4:684-687.

See Also

coalesc_abc_std(), coalesc_easyABC(), abc() in abc package, parLapply() in parallel package.

Examples

```
## Not run:
# 1/ Analysis of community assembly in a single community

# Trait-dependent filtering function
filt_gaussian <- function(t, par) exp(-(t - par[1])^2/(2*par[2]^2))

# Definition of parameters of this function and their range
par.range <- data.frame(rbind(c(0, 1), c(0.05, 1)))
row.names(par.range) <- c("topt", "sigmaopt")

# Basic summary statistics describing local community composition
f.sumstats <- function(com) array(dimnames=list(c("cwm", "cwv", "cws",
```

```
                                                    "cwk", "S", "Es")),
                                    c(mean(com[,3]), var(com[,3]),
                                      e1071::skewness(com[,3]),
                                      e1071::kurtosis(com[,3]),
                                      vegan::specnumber(table(com[,2])),
                                      vegan::diversity(table(com[,2]))))

# An observed community is simulated with known parameter values
comm <- coalesc(J = 400, m = 0.5, theta = 50,
                filt = function(x) filt_gaussian(x, c(0.2, 0.1)))

# ABC estimation of the parameters of filtering and migration based on observed
# community composition
# Number of values to sample in prior distributions
nb.samp <- 1000 # Should be large
## Warning: this function may take a while
system.time(res.single <- coalesc_abc(comm$com, comm$pool, f.sumstats = f.sumstats,
                  filt.abc = filt_gaussian, par.filt = par.range,
                  nb.samp = nb.samp, parallel = TRUE, tol = 0.5,
                  pkg = c("e1071","vegan"), method.abc = "neuralnet"))
# If the filtering function can be applied directly to a vector of trait values,
# it is strongly recommended to set "filt.vect = T"
system.time(res.single <- coalesc_abc(comm$com, comm$pool, f.sumstats = f.sumstats,
                  filt.abc = filt_gaussian, filt.vect = T, par.filt = par.range,
                  nb.samp = nb.samp, parallel = TRUE, tol = 0.5,
                  pkg = c("e1071","vegan"), method.abc = "neuralnet"))

plot(res.single$abc, param = res.single$par)
hist(res.single$abc)

# Cross validation
## Warning: this function is slow
res.single$cv <- abc::cv4abc(param = res.single$par, sumstat = res.single$ss, nval = 100,
                     tols = c(0.01, 0.1, 1), method.abc = "neuralnet")
plot(res.single$cv)

# Alternative option using a MCMC sampling approach
res.single2 <- coalesc_abc(comm$com, comm$pool, f.sumstats=f.sumstats, filt.abc=filt_gaussian,
par.filt=par.range, nb.samp=nb.samp/10, tol=0.1, type = "mcmc")

# 2/ Analysis of community assembly in multiple communities with a common pool of
# immigrants

# When the input is a site-species matrix, use argument multi="tab"
# When the input is a list of community composition, use argument multi="seqcom"

# 2.1/ Environment-dependent environmental filtering

# The variation of optimal trait values can depend on environmental variation across communities
filt_gaussian_env <- function(t, par, env) exp(-(t-par[1]*env+par[2])^2/(2*par[3]^2))
# Vector of environmental conditions across communities
env1 <- matrix(runif(20))
```

```
# Simulate a set of 20 communities with varying environmental filtering in
# different environmental conditions
# A common source pool of immigrants
pool <- coalesc(J = 10000, theta = 50)$com
meta1 <- c()
for(i in 1:20)
  meta1[[i]] <- coalesc(J = 400, pool = pool, m = 0.5,
                filt = function(x) filt_gaussian_env(x, c(0.5, 0.2, 0.1), env1[i,]))
# Build a species-by-site table
tab1 <- array(0, c(20, max(pool$sp)))
for(i in 1:20) {
  compo <- table(meta1[[i]]$com$sp)
  tab1[i, as.numeric(names(compo))] <- compo
}
colnames(tab1) <- as.character(1:ncol(tab1))
tab1 <- tab1[, colSums(tab1)!=0]

# Definition of parameters and their range
# We will estimate the slope a and intercept b of the linear relationship between
# optimal trait values and environmental conditions
par.range <- data.frame(rbind(c(-1, 1), c(0, 1), c(0.05, 1)))
row.names(par.range) <- c("a", "b", "sigmaopt")

# Basic summary statistics
# The function will provide trait-based statistics and taxonomic diversity in
# each community
f.sumstats <- function(tab, traits) array(dimnames=
 list(c(paste(rep("cwm",nrow(tab)),1:nrow(tab)), paste(rep("cwv",nrow(tab)),1:nrow(tab)),
  paste(rep("cws",nrow(tab)),1:nrow(tab)), paste(rep("cwk",nrow(tab)),1:nrow(tab)),
  paste(rep("S",nrow(tab)),1:nrow(tab)), paste(rep("Es",nrow(tab)),1:nrow(tab)),
  paste(rep("beta",nrow(tab)),1:nrow(tab)))),
  c(apply(tab, 1, function(x) Weighted.Desc.Stat::w.mean(traits[colnames(tab),], x)),
  apply(tab, 1, function(x) Weighted.Desc.Stat::w.var(traits[colnames(tab),], x)),
  apply(tab, 1, function(x) Weighted.Desc.Stat::w.skewness(traits[colnames(tab),], x)),
  apply(tab, 1, function(x) Weighted.Desc.Stat::w.kurtosis(traits[colnames(tab),], x)),
  apply(tab, 1, function(x) sum(x!=0)),
  apply(tab, 1, vegan::diversity),
  colMeans(as.matrix(vegan::betadiver(tab,"w")))))

# 2.1.1/ Using multi=tab (no local trait information)

# ABC estimation of the parameters of trait-environment relationship and of migration,
# based on observed community composition
## Warning: this function may take a while
res.tab1 <- coalesc_abc(tab1, pool, multi = "tab", f.sumstats = f.sumstats,
                  filt.abc = filt_gaussian_env, filt.vect = T, par.filt = par.range,
                  add = T, var.add = env1,
                  nb.samp = nb.samp, parallel = TRUE, tol = 0.5,
                  pkg = c("vegan","Weighted.Desc.Stat"), method.abc = "neuralnet")

plot(res.tab1$abc, param=res.tab1$par)
hist(res.tab1$abc)
```

```
# 2.1.2/ Using multi=seqcom (keeping info on local trait values)

seqcom <- lapply(meta1, function(x) x$com)

f.sumstats.seqcom <- function(seqcom) unlist(lapply(seqcom, function(x)
  c(mean(x[,3]),var(x[,3]),length(unique(x[,2])),
    vegan::diversity(table(x[,2])))))

seqcom.obs <- f.sumstats.seqcom(seqcom)

res.seqcom  <- coalesc_abc(seqcom, pool, multi = "seqcom", f.sumstats = f.sumstats.seqcom,
                                  filt.abc = filt_gaussian_env,
                                  add = T, var.add = env1,
                                  prop = F, par.filt = par.range,
                                  nb.samp = nb.samp)

# Another example with proportion data in communities

seqcom.prop <- lapply(seqcom, function(x) data.frame(
                              sp=tapply(x[,2], x[,2], function(y) y[1]),
                              cov=tapply(x[,2], x[,2], length)/nrow(x),
                              tr=tapply(x[,3], x[,2], mean)))

# Default prior for community size is uniform between 100 and 1000
# Can be changed by setting par.size

f.sumstats.seqcom.prop <- function(seqcom) unlist(lapply(seqcom, function(x)
 c(weighted.mean(x[,3], x[,2]),Weighted.Desc.Stat::w.var(x[,3], x[,2]),length(unique(x[,2])),
    sum(x[,2]^2))))

sumstats.obs <- f.sumstats.seqcom.prop(seqcom.prop)

res.seqcom.prop  <- coalesc_abc(seqcom.prop, pool, multi = "seqcom",
                                  f.sumstats = f.sumstats.seqcom.prop,
                                  filt.abc = filt_gaussian_env,
                                  add = T, var.add = env1,
                                  prop = T, par.filt = par.range,
                                  nb.samp = nb.samp)

# 2.2/ Environment-dependent environmental filtering and immigration

# In this case, the migration rate depends of another environmental variable
# representing, e.g., community isolation
# There will be two environmental variables used for parameter inference, one conditioning
# environmental filtering, and the other conditioning migration
env2 <-  matrix(runif(20))
env <- cbind(env1, env2)
colnames(env) <- c("env1", "env2")
# Migration depends on environmental conditions following some linear relationship
migr_env <- function(par, env) (par[2]-par[1])*env+par[1]

# Simulate a set of 20 communities with environment-dependent environmental filtering
# and immigration
```

```
meta2 <- c()
for(i in 1:20)
  meta2[[i]] <- coalesc(J = 400, pool = pool, m = migr_env(c(0.25,0.75), env[i,2]),
                filt = function(x) filt_gaussian_env(x, c(0.5, 0.2, 0.1), env[i,1]))
# Build a species-by-site table
tab2 <- array(0, c(20, max(pool$sp)))
for(i in 1:20) {
  compo <- table(meta2[[i]]$com$sp)
  tab2[i, as.numeric(names(compo))] <- compo
}
colnames(tab2) <- as.character(1:ncol(tab2))
tab2 <- tab2[, colSums(tab2)!=0]

# Definition of parameters and their range
# We will estimate the slope a and intercept b of the linear relationship between
# optimal trait values and environmental variable 1, and slope c and intercept d in the
# logistic function determining the variation of migration rate with environmental variable 2
par.filt.range <- data.frame(rbind(c(-1, 1), c(0, 1), c(0.05, 1)))
row.names(par.filt.range) <- c("a", "b", "sigmaopt")
par.migr.range <- data.frame(rbind(c(0, 1), c(0, 1)))
row.names(par.migr.range) <- c("c", "d")

# ABC estimation of the parameters of trait-environment and migration-environment
# relationships, based on observed community composition
## Warning: this function may take a while
nb.samp <- 200
res.tab2 <- coalesc_abc(tab2, pool, multi = "tab", f.sumstats = f.sumstats,
                  filt.abc = function(x, par, env) filt_gaussian_env(x, par, env[1]),
                  filt.vect = T, migr.abc = function(par, env) migr_env(par, env[2]),
                  par.filt = par.filt.range, par.migr = par.migr.range,
                  add = T, var.add = env,
                  nb.samp = nb.samp, parallel = FALSE, tol = 0.5,
                  pkg = c("e1071","vegan"), method.abc = "neuralnet")

plot(res.tab2$abc, param=res.tab2$par)
hist(res.tab2$abc)

# Check result consistency
comm <- 1
coeff <- summary(res.tab2$abc)
plot(cbind(sapply(pool$tra, function(x) filt_gaussian_env(x, c(0.5, 0.2, 0.1), env[comm,1])),
sapply(pool$tra, function(x) filt_gaussian_env(x, coeff[3,1:3], env[comm,1]))),
xlab = paste("Expected filter value in community ", comm, sep=" "), ylab = "Estimated filter value")
abline(0,1)
plot(cbind(sapply(env[,2], function(x) migr_env(c(0.25, 0.75), x)),
sapply(env[,2], function(x) migr_env(coeff[3,4:5], x))),
xlab = "Expected migration value", ylab = "Estimated migration value")
abline(0,1)

# 3/ Distinct pools of migrants across communities

# The immigrants can be drawn from distinct pools for each community, to represent a local context.
# In this case, the pool argument sent to coalesc_abc must be a list of the local pools.
```

```
# Simulate several communities with distinct pools and same environmental filter
pool.loc <- c()
meta3 <- c()
theta <- 2.5*(1:20) # The pools differ in diversity
# We first define a baseline pool of immigrants common to all communities
baseline <- coalesc(J = 10000, theta = 25)$com
baseline$sp <- paste(0, baseline$sp, sep = "_")
for(i in 1:20)
{
  pool.loc[[i]] <- coalesc(J = 10000, theta = theta[i])$com
  pool.loc[[i]]$sp <- paste(i, pool.loc[[i]]$sp, sep = "_")
  pool.loc[[i]] <- rbind(pool.loc[[i]], baseline)
  meta3[[i]] <- coalesc(J = 400, pool = pool.loc[[i]], m = 0.5,
                filt = function(x) filt_gaussian(x, c(0.25, 0.1)))
}
# Build a species-by-site table
tab3 <- array(0, c(20, length(unique(Reduce(rbind, pool.loc)[,2]))))
colnames(tab3) <- unique(Reduce(rbind, pool.loc)[,2])
for(i in 1:20) {
  compo <- table(meta3[[i]]$com$sp)
  tab3[i, names(compo)] <- compo
}
tab3 <- tab3[, colSums(tab3)!=0]

# ABC estimation of the parameters of trait-environment and migration-environment
# relationships, based on observed community composition
## Warning: this function may take a while
par.range <- data.frame(rbind(c(0, 1), c(0.05, 1)))
row.names(par.range) <- c("topt", "sigmaopt")
res.tab3 <- coalesc_abc(tab3, pool.loc, multi = "tab", f.sumstats = f.sumstats,
                   filt.abc = filt_gaussian, filt.vect = T,
                   par.filt = par.range,
                   nb.samp = nb.samp, parallel = FALSE, tol = 0.5,
                   pkg = c("e1071","vegan"), method.abc = "neuralnet")

plot(res.tab3$abc, param=res.tab3$par)
hist(res.tab3$abc)

## End(Not run)
```

`coalesc_abc_std` *Estimation of parameters of community assembly using Approximate Bayesian Computation (ABC) with random exploration of prior parameter distributions*

Description

Estimates parameters of neutral migration-drift dynamics (through migration rate m and parameters of environmental filtering (through a filtering function `filt.abc()`) from the composition of a local community and the related regional pool.

Usage

```
coalesc_abc_std(comm.obs, pool = NULL, multi = "single", prop = F, traits = NULL,
f.sumstats, filt.abc = NULL, filt.vect = F, migr.abc = NULL, m.replace = T,
size.abc = NULL, add = F, var.add = NULL, params = NULL, par.filt = NULL,
par.migr = NULL, par.size = NULL, constr = NULL, scale = F, dim.pca = NULL,
svd = F, theta.max = NULL, nb.samp = 10^6, parallel = TRUE, nb.core = NULL,
tol = NULL, pkg = NULL, method.abc = "rejection")

do.simul.coalesc(J, pool = NULL, multi = "single", prop = F, nb.com = NULL,
traits = NULL, f.sumstats = NULL, nb.sumstats = NULL, filt.abc = NULL,
filt.vect = F, migr.abc = NULL, m.replace = T, size.abc = NULL, add = F,
var.add = NULL, params = NULL, par.filt = NULL, par.migr = NULL, par.size = NULL,
constr = NULL, dim.pca = NULL, svd = F,  theta.max = NULL, nb.samp = 10^6,
parallel = TRUE, nb.core = NULL, pkg = NULL)

generate_prior(pool = NULL, prop = F, constr = NULL, params = NULL,
par.filt = NULL, par.migr = NULL, par.size = NULL, theta.max = NULL, nb.samp = 10^6)
```

Arguments

comm.obs	the observed community composition. If multi = "single" (default), should be a matrix or data.frame of individuals on rows with their individual id (first column), and species id (second column).
pool	composition of the regional pool to which the local community is hypothesized to be related through migration dynamics with possible environmental filtering. Should be a matrix of individuals on rows with their individual id (first column), species id (second column), and (optionally) the trait values of the individuals. When multi = "tab" or "seqcom", different pools can be set for each community: pool must be then defined as a list of matrices with the same columns as above.
multi	structure of the community inputs: • if multi = "single", comm.obs represents a single community • if multi = "tab", the user provides a site-species matrix (**sites in rows and species in columns**) • if multi = "seqcom", comm.obs contains a list of communities. If prop = F, the local community composition is given with one individual per row (first column is individual id, second column is species id, subsequent columns provide trait values). If prop = F, the local community composition is given with one species per row (first column if species id, second column is coverage, subsequent columns provide trait values).
prop	indicates if the community composition is given in term of relative species abundances, cover or proportions. In this case, a parameter of effective community size is estimated. Default is FALSE. If no prior is provided for J based on argumeng par.size, it is by default a uniform distribution between 100 and 1000.
traits	the trait values of species in the regional pool. It is used if trait information is not provided in pool. In this case, intraspecific trait variation is assumed to be null. Species names of pool must be included in traits.

`f.sumstats` a function calculating the summary statistics of community composition. It is be used to compare observed and simulated community composition in ABC estimation. The first input argument of the function concerns community composition, under a format depending on `multi` and on `prop`. If `multi="comm"` this first argument is the local community composition with individual and species labels as first columns and (optionally) trait values as subsequent columns. There can be a second argument taking the average trait values per species (derived from `pool` if not provided in `traits`, see above), and `var.add` (see below) can be sent as a third argument. See examples of different options below. The function should return a vector of summary statistics.

`nb.sumstats` number of summary statistics returned by `f.sumstats`.

`filt.abc` the hypothesized environmental filtering function. It is a function of individual trait values (first argument), additional parameters to be estimated and (optionally) environmental variables defined in `var.add`. If `NULL`, neutral communities will be simulated and only m will be estimated.

`filt.vect` indicates whether the filtering function can be vectorized. It means that the function can take as input a vector of trait values and provide a vector of the corresponding weights.

`migr.abc` a function defining immigration probability in local communities. It can be a function of environmental variables defined in `var.add` and of parameters to be estimated.

`m.replace` should the immigrants be drawn with replacement from the source pool. Default is `TRUE`.

`size.abc` a function defining local community sizes. It can be used when `prop = T`. It can be a function of environmental variables defined in `var.add` and of parameters to be estimated.

`add` indicates if additional variables must be passed to `filt.abc` and/or `migr.abc` . It can be, for instance, environmental data conditioning the trait-based filtering and/or immigration in the community. Default is FALSE.

`var.add` additional variables to be passed to `filt.abc` and/or `migr.abc` when `add = T`.

`params` equivalent to par.filt (see below): it is kept in the function for compatibility with previous versions.

`par.filt` a matrix of the bounds of the parameters used in `filt.abc`. The row names of `par.filt` provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

`par.migr` a matrix of the bounds of the parameters used in `migr.abc`. The row names of `par.migr` provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

`par.size` a matrix of the bounds of the parameters used in `size.abc`. The row names of `par.size` provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

constr	constraints that must be set on parameter values, i.e., relationships that must be met when drawing the values in prior distributions. It must be a vector of character strings each including an operator relating some of the parameters. The names of parameters must be consistent with those used in par.filt, par.migr and par.size. Default is NULL (no constraint).
scale	should the summary statistics be scaled. Default is FALSE.
dim.pca	gives a number of independent dimensions to calculate from the simulated summary statistics, by performing Principal Component Analysis (PCA). Default is NULL, in which case no PCA is performed. If svd = T, Singular Value Decomposition (SVD) will be performed instead of PCA.
svd	indicates if Singular Value Decomposition (SVD) must be performed to get the basic dimensions requested in dim.pca. Will be ignored if dim.pca = NULL.
theta.max	if pool = NULL, regional abundances will be simulated following a log-series distribution. The function will estimate the theta parameter of this distribution. theta.max then provides the upper bound for this estimation.
nb.samp	the number of parameter values to be sampled in ABC calculation. Random values of parameters of environmental filtering (see filt.abc and params) and of migration (denoted as m) are drawn from a uniform distribution between minimum and maximum values provided in params (and between 0 and 1 for m).
parallel	boolean. If parallel = TRUE, the function will perform parallel processing using the parLapply() function of package parallel. Default is parallel = FALSE.
nb.core	number of cores to be used in parallel computation if parallel = TRUE. If NULL (default), all the cores minus 1 are used. If 1, only one core is used (i.e., no parallel computing).
tol	the tolerance value used in ABC estimation (see help in abc() function of package abc for further information). Default is NULL.
pkg	packages needed for calculation of filt.abc and/or f.sumstats.
J	local community size(s). There must be several values when multi is seqcom or tab.
nb.com	number of communities.
method.abc	the method to be used in ABC estimation (see help on abc() function of package abc for further information). Can be "rejection", "loclinear", "neuralnet" or "ridge". Default is "rejection".

Details

coalesc_abc_std() provides ABC estimation of assembly parameters for one (if multi = "single", default) or several communities (if multi = "tab" or "seqcom") related to the same regional pool. It performs simulations with parameter values randomly drawn in priori distributions.

generate_prior() draws parameter values from the prior distributions defined with params,par.filt and/or par.size. do.simul.coalesc() provides simulated communities used for ABC estimation.

Value

par	parameter values used in simulations.
obs	observed summary statistics.
obs.scaled	observed summary statistics standardized according to the mean and standard deviation of simulated values.
ss	standardized summary statistics of the communities simulated with parameter values listed in par.
ss.scale	data frame including the mean and the standard deviation used for standardization of observed and summary statistics.
abc	a single (if multi = FALSE, default) or a list of abc objects including ABC estimation information for each community provided in input (comm.obs). It is of class abc for type="standard" and of class EasyABC otherwise.

Author(s)

F. Munoz, E. Barthelemy

See Also

coalesc_abc()

---

coalesc_easyABC *Estimation of parameters of community assembly using Approximate Bayesian Computation (ABC), with algorithms of optimal exploration of prior distributions*

Description

Estimates parameters of neutral migration-drift dynamics (through migration rate m and parameters of environmental filtering (through a filtering function filt.abc()) from the composition of a local community and the related regional pool.

Usage

```
coalesc_easyABC(comm.obs, pool = NULL, multi = "single", prop = F, traits = NULL,
 f.sumstats, filt.abc = NULL, filt.vect = F, migr.abc = NULL, m.replace = T,
 size.abc = NULL, add = F, var.add = NULL, params = NULL, par.filt = NULL,
 par.migr = NULL, par.size = NULL, constr = NULL, scale = F, dim.pca = NULL,
 svd = F, theta.max = NULL, nb.samp = 10^6, parallel = TRUE, nb.core = NULL,
 tol = NULL, pkg = NULL, type = NULL, method.seq = "Lenormand",
 method.mcmc = "Marjoram_original", method.abc = "rejection", alpha = 0.5)
```

Arguments

comm.obs	the observed community composition. If multi = "single" (default), should be a matrix or data.frame of individuals on rows with their individual id (first column), and species id (second column).
pool	composition of the regional pool to which the local community is hypothesized to be related through migration dynamics with possible environmental filtering. Should be a matrix of individuals on rows with their individual id (first column), species id (second column), and (optionally) the trait values of the individuals. When multi = "tab" or "seqcom", different pools can be set for each community: pool must be then defined as a list of matrices with the same columns as above.
multi	structure of the community inputs: • if multi = "single", comm.obs represents a single community • if multi = "tab", the user provides a site-species matrix (**sites in rows and species in columns**) • if multi = "seqcom", comm.obs contains a list of communities. If prop = F, the local community composition is given with one individual per row (first column is individual id, second column is species id, subsequent columns provide trait values). If prop = F, the local community composition is given with one species per row (first column if species id, second column is coverage, subsequent columns provide trait values).
prop	indicates if the community composition is given in term of relative species abundances, cover or proportions. In this case, a parameter of effective community size is estimated. Default is FALSE. If no prior is provided for J based on argumeng par.size, it is by default a uniform distribution between 100 and 1000.
traits	the trait values of species in the regional pool. It is used if trait information is not provided in pool. In this case, intraspecific trait variation is assumed to be null. Species names of pool must be included in traits.
f.sumstats	a function calculating the summary statistics of community composition. It is be used to compare observed and simulated community composition in ABC estimation. The first input argument of the function concerns community composition, under a format depending on multi and on prop. If multi="comm" this first argument is the local community composition with individual and species labels as first columns and (optionally) trait values as subsequent columns. There can be a second argument taking the average trait values per species (derived from pool if not provided in traits, see above), and var.add (see below) can be sent as a third argument. See examples of different options below. The function should return a vector of summary statistics.
filt.abc	the hypothesized environmental filtering function. It is a function of individual trait values (first argument), additional parameters to be estimated and (optionally) environmental variables defined in var.add. If NULL, neutral communities will be simulated and only m will be estimated.
filt.vect	indicates whether the filtering function can be vectorized. It means that the function can take as input a vector of trait values and provide a vector of the corresponding weights.

migr.abc — a function defining immigration probability in local communities. It can be a function of environmental variables defined in var.add and of parameters to be estimated.

m.replace — should the immigrants be drawn with replacement from the source pool. Default is TRUE.

size.abc — a function defining local community sizes. It can be used when prop = T. It can be a function of environmental variables defined in var.add and of parameters to be estimated.

add — indicates if additional variables must be passed to filt.abc and/or migr.abc . It can be, for instance, environmental data conditioning the trait-based filtering and/or immigration in the community. Default is FALSE.

var.add — additional variables to be passed to filt.abc and/or migr.abc when add = T.

params — equivalent to par.filt (see below): it is kept in the function for compatibility with previous versions.

par.filt — a matrix of the bounds of the parameters used in filt.abc. The row names of par.filt provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

par.migr — a matrix of the bounds of the parameters used in migr.abc. The row names of par.migr provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

par.size — a matrix of the bounds of the parameters used in size.abc. The row names of par.size provide the parameter names used in ABC calculation and output. First column contains minimum values and second column contains maximum values.

constr — constraints that must be set on parameter values, i.e., relationships that must be met when drawing the values in prior distributions. It must be a vector of character strings each including an operator relating some of the parameters. The names of parameters must be consistent with those used in par.filt, par.migr and par.size. Default is NULL (no constraint).

scale — should the summary statistics be scaled. Default is FALSE.

dim.pca — gives a number of independent dimensions to calculate from the simulated summary statistics, by performing Principal Component Analysis (PCA). Default is NULL, in which case no PCA is performed. If svd = T, Singular Value Decomposition (SVD) will be performed instead of PCA.

svd — indicates if Singular Value Decomposition (SVD) must be performed to get the basic dimensions requested in dim.pca. Will be ignored if dim.pca = NULL.

theta.max — if pool = NULL, regional abundances will be simulated following a log-series distribution. The function will estimate the theta parameter of this distribution. theta.max then provides the upper bound for this estimation.

nb.samp — the number of parameter values to be sampled in ABC calculation. Random values of parameters of environmental filtering (see filt.abc and params) and of migration (denoted as m) are drawn from a uniform distribution between minimum and maximum values provided in params (and between 0 and 1 for m).

parallel	boolean. If parallel = TRUE, the function will perform parallel processing using the parLapply() function of package parallel. Default is parallel = FALSE.
nb.core	number of cores to be used in parallel computation if parallel = TRUE. If NULL (default), all the cores minus 1 are used. If 1, only one core is used (i.e., no parallel computing).
tol	the tolerance value used in ABC estimation (see help in abc() function of package abc for further information). Default is NULL.
pkg	packages needed for calculation of filt.abc and/or f.sumstats.
type	the type of algorithm to be used in EasyABC. Can be either "standard" (using package abc, default)" "seq" (sequential), "mcmc" or "annealing". Three later options are based on the EasyABC package.
method.seq	when type = "seq", gives the algorithm for sequential sampling scheme, which is passed to ABC_sequential. Can be "Lenormand" (Default), "Drovandi", "Delmoral", "Beaumont" or "Emulation".
method.mcmc	when type = "mcmc", gives the algorithm for MCMC sampling scheme, which is passed to ABC_mcmc. Can be "Marjoram_original"(Default), "Marjoram" or "Wegmann". The method "Marjoram_original" cannot be used with multiple cores.
method.abc	the method to be used in ABC estimation (see help on abc() function of package abc for further information). Can be "rejection", "loclinear", "neuralnet" or "ridge". Default is "rejection".
alpha	a positive number between 0 and 1 (strictly) used when performing sequential ABC method. alpha is the proportion of particles rejected at each step in the algorithm "Drovandi". This is the proportion of particles kept at each step in the algorithms "Delmoral", "Lenormand" and "Emulation". Default value is 0.5

Details

coalesc_easyABC() provides ABC estimation of assembly parameters for one (if multi = "single", default) or several communities (if multi = "tab" or "seqcom") related to the same regional pool. The function uses optimization algorithms implemented in the package EasyABC.

Value

par	parameter values used in simulations.
obs	observed summary statistics.
obs.scaled	observed summary statistics standardized according to the mean and standard deviation of simulated values.
ss	standardized summary statistics of the communities simulated with parameter values listed in par.
ss.scale	data frame including the mean and the standard deviation used for standardization of observed and summary statistics.
abc	a single (if multi = FALSE, default) or a list of abc objects including ABC estimation information for each community provided in input (comm.obs). It is of class abc for type="standard" and of class EasyABC otherwise.

Author(s)

F. Munoz, E. Barthelemy

References

Jabot, F., T. Faure, and N. Dumoulin 2013. EasyABC: performing efficient approximate Bayesian computation sampling schemes using R. Methods in Ecology and Evolution 4:684-687.

See Also

coalesc_abc()

---

forward — *Simulation of neutral and niche-based community dynamics forward in time*

Description

Simulates niche-based (habitat filtering and/or limiting similarity) and neutral community dynamics from a given initial composition, over a given number of generations.

Usage

```
forward(initial, m = 1, theta = NULL, d = 1, gens = 150, keep = FALSE,
       pool = NULL, traits = NULL, filt = NULL, filt.vect = F, limit.sim = NULL,
        limit.intra = F, par.limit = 0.1, coeff.lim.sim = 1,
        type.filt = "immig", type.limit = "death", add = F, var.add = NULL,
      prob.death = NULL, method.dist = "euclidean", checks = T, plot_gens = FALSE)
gauss_limit(dist, par)
get_number_of_gens(given_size, pool, traits = NULL, nbrep = 5, m = 1, theta = NULL,
                   d = 1, gens = NULL, filt = NULL,  limit.sim = NULL,
                   par.limit = 0.1, coeff.lim.sim = 1, type.filt = "immig",
                  type.limit = "death", m.replace = T, add = F, var.add = NULL,
                   prob.death = NULL, method.dist = "euclidean",
                   plot_gens = FALSE)
pick(com, d = 1, m = 1, theta = NULL, pool = NULL, prob.death = NULL,
    filt = NULL, filt.vect = F, limit.sim = NULL,  limit.intra = F, par.limit = 0.1,
     coeff.lim.sim = 1, type.filt = "immig", type.limit = "death",
   m.replace = T, add = F, var.add = NULL, new.index = 0, method.dist = "euclidean")
pick.mutate(com, d = 1, mu = 0, new.index = 0)
pick.immigrate(com, d = 1, m = 1, pool, prob.death = NULL,
              filt = NULL, filt.vect = F, limit.sim = NULL,  limit.intra = F,
              par.limit = 0.1, coeff.lim.sim = 1,
              type.filt = "immig", type.limit = "death", m.replace = T,
              add = F, var.add = NULL, method.dist = "euclidean")
```

Arguments

com, initial	starting community. It is in principle a three (or more) column matrix or data.frame including individual ID, species names and trait values. For strictly neutral dynamics, it can be a vector of individual species names.
m	migration rate (if m = 1 the community is a subsample of the regional pool).
theta	parameter of neutral dynamics in the regional pool (used only if pool=NULL), it is the "fundamental biodiversity number" (θ).
mu	mutation rate derived from θ, such as $\mu = \theta/(2 * J_m)$, where J_m is the size of the metacommunity.
d	number of individuals that die in each time-step.
gens	number of generations to simulate.
keep	boolean value. If FALSE (default) the function output only the community composition at the end of the simulation. If TRUE the function output a list of community composition at successive time steps (see Value section).
pool	the regional pool of species providing immigrants to the local community. It is in principle a three-column matrix or data frame including individual ID, species names and trait values. If trait information is missing, a random trait value is given to individuals, from a uniform distribution between 0 and 1. If NULL, the pool is simulated as a metacommunity at speciation-drift equilibrium, based on prob for speciation rate.
traits	a matrix or data.frame including one or several traits on columns. A unique trait value is assigned to each species in the regional pool. Species names of pool must be included in rownames of traits. If traits = NULL and trait information is absent from pool, a random trait value is given to species of the regional pool, from a uniform distribution between 0 and 1.
given_size	size of the community you want to have an estimate of the number of generations needed to reach stationarity in species richness.
nbrep	number of replicates from which you want to estimate the number of generations needed to reach stationarity in species richness.
limit.sim	if non null, limiting similarity will be simulated, based on species trait distances (computed with the method given by method.dist) and a Gaussian overlapping function.
limit.intra	should limiting similarity play on individual dynamics within species? Default is FALSE.
par.limit	a vector of additional parameters to be passed to the limiting similarity function defined in limit.sim. If NULL but the limit.sim requires additional parameter values, an error message will be issues. In the case of default Gaussian limiting similarity function, the additional parameter value sets the standard deviation of the Gaussian function.
coeff.lim.sim	adjust the intensity of limiting similarity.
type.filt	indicates how habitat filtering plays in community assembly. If type.filt="death", it plays on mortality events. If type.filt="immig" (default), it plays at the establishment of immigrants, If type.filt="loc.recr", it plays at the establishment of local offspring. type.filt can be any combination of these three values.

type.limit	indicates how limiting similarity plays in community assembly. If type.limit="death" (default), it plays on mortality events. If type.limit="immig", it plays at the establishment of immigrants, If type.limit="loc.recr", it plays at the establishment of local offspring. type.limit can be any combination of these three values.
filt	the function used to represent habitat filtering. For a given trait value t, filt(t) represents the probability that an individual with trait t enters the local community.
filt.vect	indicates whether the filtering function can be vectorized. It means that the function can take as input a vector of trait values and provide a vector of the corresponding weights.
m.replace	should the immigrants be drawn with replacement from the source pool. Default is TRUE.
add	indicates if additional variables must be passed to filt. It can be, for instance, environmental data conditioning the trait-based filtering in the community. Default is FALSE.
var.add	additional variables to be passed to filt when add = T.
prob.death	provides a baseline probability of death that is homogeneous across species. It is used in niche-based dynamics to represent the balance of baseline and niche-dependent mortality.
method.dist	provides the method to compute trait distances between individuals (syntax of function dist, can be in the list c("euclidean","maximum", "manhattan","canberra", "binary", "minkowski")).
new.index	prefix used to give a new species name when speciation occurs.
plot_gens	plot the number of unique individuals and species over generations.
checks	should initial checks that the inputs are correct be performed?
dist	vector of trait distances that is used for calculation of limiting similarity.
par	addditional parameter values to be passed to limit.sim function.

Details

The model of community assembly is a zero-sum game, so that the number of individuals of the community is fixed to the number of individuals in initial community.

Two types of niche-based processes, habitat filtering and limiting similarity can affect (i) immigration, (ii) mortality, and (iii) recruitment of local offspring. The user can define on each of the three components the processes play, by defining type.filt and type.limit. These parameters can combine one of several values in "immig", "death" and "loc.recr" depending on either the process plays on immigration, (ii) mortality, and (iii) recruitment of local offspring, respectively. For instance, defining type.filt = c("immig", "death") would mean that habitat filtering plays on both immigration and on death events. By default, type.filt = "immig" and type.simil = "death", which corresponds to a classical conception of the hierarchical influence of habitat filtering and limiting similarity on immigration success and local survival.

A environmental filtering function can be defined with filt. It should take two arguments, the first is a trait value of a candidate immigrant, the second is a vector including the parameter values of

the filtering function. See examples below for further information on usage. An important point is that in many cases the function might be vectorized, that is, it can provide a vector of filtering probabilities for a vector of trait values given in first argument. In this case the user should set `filt.vect = T`, which will significantly accelerate simulations.

`gauss_limit()` is the default function used to compute limiting similarity. In this case, the relative performance of an individual decreases depending on the sum of exponential distances to other individuals in the community (McArthur and Levins 1967), i.e.,

Function `get_number_of_gen()` allows determining the number of generations needed to reach stationary richness for given parameterization of `forward()`. The target number of generation is based on assessing the change point in species richness change over time for replicate simulated communities with random initial composition. A conservative measure is proposed as the maximum time to reach stationary richness over the replicate simulated communities.

Functions `pick.immigrate()` and `pick.mutate()` are used to simulate immigration and speciation events within a time step. They are embedded in forward and are not really intended for the end user.

A Shiny app is available to visualize simulated trait distributions for chosen parameter values in the model.

Value

`com`	if `keep = FALSE`, a data.frame of simulated individuals, with the label of ancestor individual in the regional pool on first column (as in the first column of the pool), species label on second column (as in the second column of the pool), and species trait (as in the third column of the pool).
`pool`	a data.frame of the individuals of the regional source pool, with the label of ancestor individual in the regional pool on first column (as in first column of input `pool`), species label on second column (as in second column of input `pool`), and species trait (as in third column of input `pool`).
`sp_t`	a vector of species richness at each time step.
`com_t`	if `keep = TRUE`, a list of community composition for each time step (a data.frame as in `com`).
`dist.t`	if `limit.sim = TRUE`, the average value of the limiting similarity function over time.
`new.index`	for `pick.mutate()`, return the new index to be used for species name at a next speciation event.
`call`	the call function.

Author(s)

F. Munoz, derived from the `untb` function of R. Hankin.

References

For neutral dynamics, see S. P. Hubbell 2001. "The Unified Neutral Theory of Biodiversity". Princeton University Press.

For the default model of limiting similarity, MacArthur, R., Levins, R., 1967. The limiting similarity, convergence, and divergence of coexisting species. The American Naturalist 101, 377–385.

Examples

```
## Not run:
# Initial community composed of 10 species each including 10 individuals
initial1 <- rep(as.character(1:10), each = 10)

# Simulation of speciation and drift dynamics over 1000 time steps
final1 <- forward(initial = initial1, theta = 50, gens = 1000)
# The final community includes new species (by default names begins with "new")
final1$com$sp # includes new species generated by speciation events

# A regional pool including 100 species each including 10 individuals
pool <- rep(paste("pool.sp",as.character(1:100), sep=""), each = 10)

# Simulation of migration and drift dynamics over 1000 time steps
final2 <- forward(initial = initial1, m = 0.1, gens = 1000, pool = pool)
# The final community includes species that have immigrated from the pool
final2$com$sp # includes new species that immigrated from the pool

# Initial community composed of 10 species each including 10 individuals,
# with trait information for niche-based dynamics
initial2 <- data.frame(ind = 1:100, sp = rep(as.character(1:10), each = 10),
                       trait = runif(100), stringsAsFactors = F)

# Simulation of stabilizing hab. filtering around t = 0.5, over 2000 time steps
sigm <- 0.1
filt_gaussian <- function(t,x) exp(-(x - t)^2/(2*sigm^2))
# Filtering only plays only on immigration (default type.filt = "immig")
final3 <- forward(initial = initial2, m = 0.1, gens = 2000, pool = pool,
                  filt = function(x) filt_gaussian(0.5,x))
plot_comm(final3) # trait distribution in final community
# Filtering also plays on local mortality and recruitment
final4 <- forward(initial = initial2, m = 0.1, gens = 2000, pool = pool,
                  filt = function(x) filt_gaussian(0.5,x),
                  type.filt = c("immig", "death", "loc.recr"))
plot_comm(final4) # Stronger convergence around 0.5

# Simulation of stabilizing hab. filtering with two traits
pool <- data.frame(ind=1:10000, sp=rep(1:500, 20), tra1=rep(NA, 10000), tra2=rep(NA, 10000))
# Trait values in the pool
t1.sp <- runif(500)
t2.sp <- runif(500)
pool[, 3] <- t1.sp[pool[,2]]
pool[, 4] <- t2.sp[pool[,2]]
initial3 <- pool[sample(1:10000, 100),]
# Parameters of filtering for first trait
topt1 <- 0.25
sigm1 <- 0.1
# Parameters of filtering for second trait
topt2 <- 0.75
sigm2 <- 0.2
filt_gaussian <- function(x) exp(-(x[1] - topt1)^2/(2*sigm1^2)) * exp(-(x[2] - topt2)^2/(2*sigm2^2))
system.time(final5 <- forward(initial = initial3, m = 0.1, gens = 2000, pool = pool,
```

```
                filt = filt_gaussian))
# Here the filtering function can be vectorized
system.time(final5 <- forward(initial = initial3, m = 0.1, gens = 2000, pool = pool,
                filt = filt_gaussian, filt.vect = T))
plot_comm(final5, seltrait = 1) # trait distribution in final community (first trait)
plot_comm(final5, seltrait = 2) # trait distribution in final community (second trait)

# Simulation of limiting similarity, over 2000 time steps
final6 <- forward(initial = initial2, m = 0.1, gens = 2000, pool = pool,
                limit.sim = TRUE)
plot_comm(final6)
# Check temporal changes
plot(final6$sp_t, xlab = "Time step", ylab = "Community richness")
# Index of limiting similarity over time
plot(final6$dist.t, xlab = "Time step", ylab = "Limiting similarity")

# Higher migration rate, 5000 time steps
# Limiting similarity only plays on mortality
final7 <- forward(initial = initial2, m = 0.7, gens = 5000, pool = pool,
                limit.sim = TRUE)
plot_comm(final7) # should be closer to regional distribution
# Limiting similarity plays on immigration, mortality and local recruitment
final8 <- forward(initial = initial2, m = 0.7, gens = 5000, pool = pool,
                limit.sim = TRUE, type.limit = c("immig", "death", "loc.recr"))
plot_comm(final8)
# Check temporal changes
plot(final8$sp_t, xlab = "Time step", ylab = "Community richness")
points(1:5000, final7$sp_t, col="blue")
# Index of limiting similarity over time
plot(final8$dist.t, xlab = "Time step", ylab = "Limiting similarity")
points(1:5000, final7$dist.t, col="blue")

## End(Not run)
```

`plot_comm` *Plotting trait and species distributions of simulated communities*

Description

Graphical function to used on the output of `coalesc()` or `forward()` functions. It can show the links between regional and local trait/abundance distributions, or local species and rank abundance distributions.

Usage

```
plot_comm(x, type = "trait", seltrait = 1, main = NULL)
```

Arguments

x	a list including the species pool composition (x$pool) and the local community composition (x$com). For example, x may be the output of coalesc() or forward() functions.
type	• if type = "trait", the function displays density plots of trait distributions. • if type = "locreg",it displays the relationship between local and regional abundances. • if type = "sad", it displays the empirical percentiles and theoretical values of species abundance distribution (SAD) in the community. • if type = "rad", it displays the empirical percentiles and theoretical values of rank abundance distribution (RAD) in the community.
seltrait	index of the trait to be plotted following community data.frame (if multiple traits used in simulation).
main	an overall title for the plot.

Details

If type = "trait", the function provides density plots of the trait or abundance distributions in the regional pool and in a local community. If type = "locreg", it displays the relationship between regional and local species relative abundances. If type = "sad" or "rad", it shows the percentile plots provided by functions in the "sads" package (see ppsad). The reference red line corresponds to log-series for "sad" and to geometric series for "rad".

By default type = "trait".

To be used on the output of coalesc() or forward() functions.

Value

Return two stacked ggplot2 density plots if type = "trait" and biplots otherwise.

Author(s)

F. Munoz; P. Denelle

Examples

```
# Simulation of a neutral community including 100 individuals
J <- 500; theta <- 50; m <- 0.1;
comm1 <- coalesc(J, m, theta)
plot_comm(comm1)
plot_comm(comm1, type = "locreg")
plot_comm(comm1, type = "sad")

# Stabilizing habitat filtering around t = 0.5
filt_gaussian <- function(x) exp(-(x - 0.5)^2/(2*0.1^2))
comm2 <- coalesc(J, m, theta, filt = filt_gaussian)
plot_comm(comm2)
plot_comm(comm2, type = "locreg")
plot_comm(comm1, type = "sad")
```

prdch4coalesc *Posterior predictive checks using coalescent-based simulation of ecological communities*

Description

This function performs posterior predicitve checks assessing a model's prediction performance of either user defined summary statistics, functional diversity (intrspecific trait distributions, moments of the trait distribution, Enquist et al., 2015) or species rank-abundance plots

Usage

```
prdch4coalesc(com.obs, pool, filt, params, stats = "abund", f.stats = NULL,
              estim = NULL, nval = 100, progress = TRUE)
```

Arguments

com.obs	a data.frame of observed individuals, the first column should contain individual labels, the second the names of the species and the third species trait values - should be the same as provided for the ABC analysis in coalesc_abc
pool	a data.frame containing the regional pool of species providing immigrants to the local community. It should include the label of individual on first column, its species on second column and their trait values in the third column as in com.obs - shuld be the same as provided for the ABC analysis in coalesc-abc
filt	the inferred environmental filtering function. If filt = NULL predicitive checks will be performed for a purely neutral model and params should only be one column containing the posterior distribution of the migration rate parameter m
params	the posterior distribution resulting from the ABC analysis, should be a data.frame containing each parameter distribution as columns (posterior distribution of the migration rate parameter (m) should be the last column)
stats	statistics used for the predictive checks: • if stats = "custom", predictive checks using a user-defined function provided in f.stats • if stats = "moments", predictive checks using the 4 first moments of the trait distribution evaluated by comparing the observed moments to those of 100 simulated communities under the model of interest • if stats = "abund", returns the confidence interval for the rank-abundance curves simulated by the model as well as the observed species rank-abundance curve distinguishing between species whose relative abundance is either over or under-estimated by the model • if stats = "intra", returns the names of species whose trait distributions are either over or under-estimated by the model by comparing the observed mean trait value per species to their simulated counterparts
f.stats	the user-defined function computing the summary statistics to be used in the predictive checks when stats = "custom"

estim	an estimator of the posterior distribution - can be either "mean", "median" or "mode" used for the simulations
nval	the number of simulations to be run for computing predictive checks. If "estim = NULL" (default), a sample of size nval is used, if an estimator of the posterior distribution is used then nval replicates are used
progress	whether to display progress bar (default is TRUE)

Value

pvalue	probabilities that the observed user-defined statistics are greater than the same statistics simulated by the model (if stats="custom")
prd.moments	probabilities that the first four moments of the observed trait distribution are greater than those simulated by the model (if stats="moments")
underestim.sp	the names of the under-represented species by the model evaluated by comparing observed relative abundances of species to those simulated by the model (if stats="abund")
overestim.sp	the names of over-represented species by the model evaluated by comparing observed relative abundances of species to those simulated by the model (if stats="abund")

Also, if stats="abund", rank-abundance curves are displayed distinguishing between species whose relative abundance is either over or under-estimated by the model as well as the confidence interval of the rank-abundance curves derived from the simulated communities

Author(s)

E. Barthelemy

References

Enquist, Brian J., et al. "Scaling from traits to ecosystems: developing a general trait driver theory via integrating trait-based and metabolic scaling theories." Advances in ecological research. Vol. 52. Academic Press, 2015. 249-318.

Csillery, Katalin, et al. "Approximate Bayesian computation (ABC) in practice." Trends in ecology & evolution 25.7 (2010): 410-418.

Examples

```
pool <- data.frame(ind = 1:1000,
                   sp = as.character(rep(1:50), each = 10),
                   trait = runif(1000), stringsAsFactors = FALSE)
com.obs <- pool[sample(nrow(pool), 50),]

f.stats <- function(com){
  tab <- table(com[,2])
  as.vector(t(sapply(0:3, function(x) hillR::hill_taxa(tab, q=x)))) }

filt <- function(x, par) exp(-(x - par[[1]])^2/(2*par[[2]]^2))
```

```
stats <- c("custom","abund", "moments")

params <- data.frame(par1 = runif(100,0,1),
                     par2 = runif(100, .5, .9),
                     par3 = runif(100,0,1))

checks <- prdch4coalesc(com.obs, pool, filt, params, stats, f.stats)
```

`sumstats` *Predifined summary statistic function for estimating neutral and non-neutral parameters of community assembly using Approximate Bayesian Computation (ABC) implemented in coalesc_abc2.*

Description

Adaptable statistic function to be provided into `coalesc_abc2()` to estimate parameters of neutral migration-drift dynamics and parameters of environmental filtering using ABC and adapted to the format imput of the local community. These return a set of diversity indexes which can be based on species local abundances, functional traits or both.

Usage

```
sumstats(com, multi="single", traits = NULL, type = "mix", n = 4, m = 4)
```

Arguments

`com`	the observed community composition. If `multi = "single"` (default), should be a matrix or data.frame of individuals on rows with their individual id (first column), species id (second column) and trait information (optional third column)
`multi`	structure of the community input: • if `multi = "single"`, `comm.obs` contains a single community. • if `multi = "tab"`, the user provides a site-species matrix (**sites in rows and species in columns**) • if `multi = "seqcom"`, `comm.obs` contains a list of communities (not yet)
`traits`	trait values (one or several traits in column) for the species present in `com`.
`type`	Type of diversity indices to be included in the summary statistics: • if `type = "taxo"`, summary statistics are the `n` first Hill diversity numbers • if `type = "func"`, summary statistics are the 4 first moments of the trait distribution, corresponding to the community weighted mean, variance, skewness and kurtosis. • if `type = "mix"` (default),summary staitstics include the first 4 moments of the trait distribution (cwm, cwv, cws, cwk) as well as the first 4 Hill diversity numbers.
`n`	number of Hill's diversity numbers to compute when `type` is taxo or mix. Default is `n=4`.

m number of trait distribution moments to compute when type is func or mix. Default is m=4.

Details

sumstats provides summary statistics based on user preference and the format of community data to be used in coalesc_abc2 for investigating neutral or niche-based dynamics in community assembly using ABC.

Value

A vector of summary statistics to be used to compare observed and simulated community composition in the ABC estimation performed by coalesc_abc2.

For community information of the type multi = "single" this vector is comprised of n Hill numbers (when type is "taxo", the m first moments of the trait distribution when type is "func" or the m first moments of the trait distribution followed by the n Hill numbers by when type is "mix" of the single community.

For community information as a species by site matrix or data.frame with X sites, this vector is comprised of n Hill numbers of the first community, followed by the n Hill numbers for the second and so on until X (when type is "taxo"). Likewise (when type is "func", with m first moments of the trait distributions of the X communities. When type is "mix", the vector gives the m first moments of the trait distribution for every community followed by their n Hill numbers.

Author(s)

E.Barthelemy

References

Ref?

Examples

```
  ## Not run:
    examples?

## End(Not run)
```

---

tcor *Generates Correlated Traits*

Description

Create two random vectors of traits correlated between each other or a vector of traits correlated to an existing one. The linear correlation is defined by the parameter rho.

Usage

```
tcor(n, rho = 0.5, mar.fun = rnorm, x = NULL, ...)
```

Arguments

n	the integer number of values to be generated.
rho	a numeric parameter defining the linear correlation between the two traits (default is 0.5). It must belong to the interval [-1, 1].
x	an vector of numeric values. Default is NULL.
mar.fun	a function defining the random generation for the trait distribution. Default is rnorm.
...	other arguments for the mar.fun() function.

Details

rho parameter is set to 0.5 by default. x = NULL by default. Code adapted from: http://stats.stackexchange.com/questions/15011/generate-a-random-variable-with-a-defined-correlation-to-an-exis

Value

Return a data.frame with two numeric columns, each column defining a trait.

Author(s)

P. Denelle F. Munoz

Examples

```
# With no predefined trait
traits <- tcor(n = 10000, rho = 0.8)
plot(traits[, 1], traits[, 2])
cor(traits[, 1], traits[, 2])

# With existing trait
existing_trait <- rnorm(10000, 10, 1)
traits <- tcor(n = 10000, rho = 0.8, x = existing_trait)
plot(traits[, 1], traits[, 2])
cor(traits[, 1], traits[, 2])
```

Index