Package ‘untb’

March 3, 2019

Type Package
Title Ecological Drift under the UNTB
Version 1.7-4
Depends R (>= 2.10)
SystemRequirements PARI/GP >= 2.3.0 [strongly recommended for logkda()]
Imports Brobdingnag (>= 1.1-8), partitions (>= 1.9-14), polynom
Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>
Description Hubbell's Unified Neutral Theory of Biodiversity.
License GPL
URL https://github.com/RobinHankin/untb.git
BugReports https://github.com/RobinHankin/untb/issues
NeedsCompilation no
Author Robin K. S. Hankin [aut, cre] (<https://orcid.org/0000-0001-5982-0415>)
Repository CRAN
Date/Publication 2019-03-03 21:00:11 UTC

R topics documented:

  untb-package .................................................. 2
  +.count .......................................................... 3
  alonso ........................................................... 4
  bci ............................................................... 5
  butterflies ....................................................... 6
  caruso ............................................................ 6
  census ............................................................ 8
  copepod ........................................................... 9
  count ............................................................. 10
  display.untb ..................................................... 11
  etienne .......................................................... 12
Descriptive statistics and visualizations of the unified neutral theory of biodiversity.

**Details**

- **Package:** untb
- **Type:** Package
- **Version:** 1.0
- **Date:** 2005-10-26
- **License:** GPL
Package untb uses two classes of object to represent an ecosystem: class count and class census. In essence, a count object is a table of species abundances and a census object is a list of individuals. See ?census and ?count for more details. Although objects of either class can be coerced to the other, class count is the preferred form: it is a more compact representation, especially for large ecosystems.

The package simulates neutral ecological drift using function untb(). Function display.untb() displays a semi-animated graphic of an ecosystem undergoing neutral drift.

Author(s)
Robin K. S. Hankin
Maintainer: <hankin.robin@gmail.com>

References

Examples

a <- untb(start=rep(1,100),prob=0.005,gens=5000,keep=FALSE)
preston(a)
no.of.spp(a)

display.untb(start=rep(1,100),prob=0.1,gens=1000)

data(butterflies)
plot(butterflies,uncertainty=TRUE)

---

+.count

Add two count objects

Description

Adds two count objects

Usage

## S3 method for class 'count'
a + b

## S3 method for class 'census'
a + b

Arguments

a,b objects of class count
Details

Consider count objects `a` and `b`. Then `a + b` is a count object that records the number of each species in `a` and `b` combined. It is as though the organisms in the surveys were pooled.

Census objects are coerced to count objects, added, then the result coerced to a count object.

The operation is commutative and associative.

Author(s)

Robin K. S. Hankin, based on an R-help tip from Gabor Grothendieck

Examples

```r
a <- count(c(dogs=4, pigs=0, slugs=5))
b <- count(c(slugs=4, hogs=1, frogs=19))
a + b
```

Description

Various functions from Alonso and McKane 2004 dealing with analytical solutions of a neutral model of biodiversity

Usage

```r
alonso.eqn6(JM, n, theta)
alonso.eqn11(J, n, theta)
alonso.eqn12(J, n, theta, give=FALSE)
```

Arguments

- `J, JM` Size of the community and metacommunity respectively
- `n` Abundance
- `theta` Biodiversity constant
- `give` In function `alonso.eqn12()`, Boolean with default `FALSE` meaning to return the value of the integral, and `TRUE` meaning to return the full output of `integrate()`

Details

Notation follows that of Alonso and McKane 2004

Note

Function `alonso.eqn6()` is identical to function `vallade.eqn5()`
Author(s)

Robin K. S. Hankin

References


Examples

```r
J <- 100
plot(1:J, alonso.eqn11(J,n=1:J, theta=5),log="y",type="l",xlab="n",ylab=expression(S(n)),main="Eqns 11 and 12 of Alonso and McKane")
points(1:J, alonso.eqn12(J,n=1:J, theta=5),type="l",lty=2)
legend("topright",legend=c("equation 11","equation 12"),lty=1:2)
```

bci

Barro Colorado Island (BCI) dataset

Description

The BCI dataset contains location and species identity for all 10cm dbh (diameter at breast height) trees on Barro Colorado Island, currently for years 1981-1983, 1985, 1990, 1995, 2000, and 2005. The subset of interest here is the abundances for each of the 252 species recorded.

The BCI dataset is not included in the `untb` package, because its licence appears to be inconsistent with the GPL.

It is discussed here because it was used as an example dataset in Hankin 2007.

Source


References

**butterflies**

*abundance data for butterflies*

**Description**

A dataset of class “count” showing the abundance of several butterfly species

**Usage**

```r
data(butterflies)
```

**Format**

A table with names of different butterfly species, and entries corresponding to the respective numbers of individuals.

**References**

Texas Birding and Naturalist Web

**Examples**

```r
data(butterflies)
plot(butterflies, uncertainty=TRUE)
```

---

**caruso**

*Dataset due to Caruso*

**Description**

A dataframe in standard format due to Migliorini and Caruso presenting observations of oribatid mites.

**Usage**

```r
data(caruso)
```

**Format**

Dataset **caruso** is a data frame with 194 observations on 5 variables. Each row corresponds to a species; the observations (rows) are the species abundances in each of 5 habitats.

Following Migliorini et al 2002, the habitats were:

- a pure beech woodland (‘Beech’) 
- a coppice woodland (‘Coppice’)
Details

Oribatid mites are rather small and very interesting free living soil microarthropods. They have a huge species diversity with populations characterised by highly aggregated distributions over multiple spatial scales ranging from a few centimetres to hundreds of meters.

Within each habitat, several soil samples were collected (five randomly located replicates per each month: see the paper Migliorini et al. 2002). So, actually, that is a network of small samples that make a single large sample.

The five study areas of this data set belong to five habitats that are very typical of that Mediterranean region. These five areas also belong to a rather homogeneous biogeographical region (southern Tuscany). On the ground of what is known on the biology and community patterns of Oribatida, several a-priori hypotheses can be made on expected changes in the diversity of their assemblages and immigration rates respectively between and within the five areas. For instance, under the Neutral Model one might expect that the Beech forest should have the highest Theta and an immigration rate of about 1, while one might expect the opposite for the Biancana (a very arid habitat, a kind of gariga/garrigue with very patchy vegetation).

Note

Executing optimal.params.sloss(caruso) does not return useful output. The reason for this is unknown.

Source

Data kindly supplied by Tancredi Caruso

References


See Also

extractor

Examples

data(caruso)

summary(count(caruso[,1]))
census

Construct, coerce, and test for a census object

Description

In package untb, ecosystem data is held in one of two preferred forms: census data and count data. Function as.census() coerces to census format.

Usage

census(a)
as.census(a)
is.census(a)

Arguments

a Ecosystem data. In function as.census(), if a table, interpret as species count data; otherwise, interpret as census data

Details

A “census” object is a list of individuals in the form of an unnamed vector whose elements indicate the individuals’ species; compare “count” objects.

An object of class “census” is also an unordered factor. The levels are always in alphabetical order.

Function census() takes an object of class “count” and returns an object of class “census”. This function is not really intended for the end user.

Function as.census() coerces to class “count” then returns census() of the result.

Value

Returns an object of class “census”.

Author(s)

Robin K. S. Hankin

See Also

count
copepod

Examples

jj <- c(dogs=4,pigs=10,slugs=0,fish=1)

x <- census(jj)  # slugs appear as zero abundance
extant(x)  # slugs gone

x+x  # count method for census objects: order of elements lost

as.census(jj)  # probably NOT what you meant

a <- c(rep("oak",5),rep("ash",2),rep("elm",3),rep("xx",4))
# note that "a" is a plain vector here.

as.census(a)

copepod  Copepod data supplied by Phil Pugh

Description

A dataset of copepod (resp: ostracod) abundances supplied by Dr Phil Pugh of the National Oceanography Centre, Southampton

Usage

data(copepod)
data(ostracod)

Format

A table with names of different copepod (resp: ostracod) species, and entries corresponding to the numbers of individuals of each species.

Source

Kindly supplied by Southampton Oceanography Centre.

Examples

data(copepod)

optimize(f=theta.likelihood,interval=c(0,100), maximum=TRUE,
S=no.of.spp(copepod), J=no.of.ind(copepod), give.log=TRUE)

data(ostracod)

preston(ostracod)
Construct, coerce, and test for a count object

Description

In package untb, ecosystem data is held in one of two preferred forms: census data and count data. Function count creates an object of class “count”, and as.count() coerces to this class.

Usage

as.count(a, add="")
count(a)
is.count(a)

Arguments

a Ecosystem data. In function as.count(), if a table, interpret as species count data; otherwise, interpret as census data. Special dispensation is made for single rows of a dataframe

add In function as.count(), character argument with default "" (empty string) meaning to leave the species names unchanged. A non-empty string is prepended to the species names using paste(). This is useful if the species names are integers because the display can become confusing.

Details

A “count” object is a list of species together with their abundance. It also has class “table”; compare “census” objects.

An object of class “count” is a table sorted from most to least abundant species. The singletons are thus tabulated last.

Function count() takes a vector, the elements of which are interpreted as abundances. If any of the elements are named, the names are interpreted as species names (unnamed elements are given the null name). If the vector is unnamed, then the species names are upper case letters, with the first element being named “A”, the second “B”, and so on; this behaviour is inherited from as.table(). Note that this means that the species names are not necessarily in alphabetical order. From version 1.6-9, zero elements are interpreted as zero abundance species (ie extinct).

To access or change species names, use names() and names<- respectively.

Function as.count() coerces its argument to count form.

Value

Returns an object of class “count”.

Author(s)

Robin K. S. Hankin
display.unTB

See Also
census

Examples

```r
a <- c(rep("oak",5),rep("ash",2),rep("elm",3),rep("xx",4))
as.count(a)

data(saunders)
as.count(saunders[1,-(1:150)])

jj <- sample(1:5,5,replace=TRUE)
as.count(jj)
as.count(jj,add="spp.")
```

---

display.unTB  

Animation of neutral ecological drift

Description

Displays an ongoing simulation of neutral ecological drift using nice colours and a simple animation technique

Usage

```r
display.unTB(start, gens=100, prob.of.mutate = 0, cex=3, individually = TRUE, ask = FALSE, flash = FALSE, delay = 0, cols=NULL, ...)
```

Arguments

- **start**: Starting ecosystem; coerced to class census. Usually, pass an object of class count; see examples. To start with a monoculture of size 10, use `start=rep(1,10)` and to start with a system of maximal diversity (ie all singletons), use `start=1:10`
- **gens**: Number of generations to simulate
- **prob.of.mutate**: Probability of mutation. The default of zero corresponds to $\theta = 0$ and this means that any ecosystem will eventually become a monoculture (it is particularly instructive to watch this happen, especially with a starting ecosystem of maximal diversity—but be warned, this can take a long time, especially for ecosystems with a large number of individuals). Nonzero values mean that a nontrivial dominance-diversity curve will eventuate, although this too can take a long time
to happen. Try a nonzero value of prob.of.mutate with monoculture start (use individually=FALSE for such experiments)

cex The size of the dots used for plotting, defaulting to 3

individually Boolean, with default TRUE meaning that a timestep means the death of a single individual and the simultaneous birth of a new individual; and FALSE meaning that a timestep refers to every individual in the system

ask Boolean, with default FALSE meaning to display the generations autonomously, and TRUE meaning to wait for the user to hit the “return” before proceeding

flash Boolean, with TRUE meaning to indicate the site of a death/birth with a flashing ring; and default FALSE meaning to omit the flashing ring. Use TRUE for pedagogic purposes, possibly with ask set to TRUE, or a nonzero delay. This option only kicks in if individually is TRUE

delay Time delay between generations in seconds; meaningful whatever the value of flash and individually

cols A vector of colours with default NULL meaning to choose them randomly. Useful for printing stills from a movie

... Further arguments passed to plot() and points()

Author(s)

Robin K. S. Hankin

References


Examples

data(butterflies)
display.untb(start=butterflies,prob=0, gens=1e2)

Description

Function etienne() returns the probability of a given dataset given theta and m according to the Etienne’s sampling formula. Function optimal.params() returns the maximum likelihood estimates for theta and m using numerical optimization

Usage

etienne(theta, m, D, log.kda = NULL, give.log = TRUE, give.like = TRUE)
optimal.params(D, log.kda = NULL, start = NULL, give = FALSE, ...)

Etienne’s sampling formula
etienne

Arguments

theta     Fundamental biodiversity parameter
m         Immigration probability
D         Dataset; a count object
log.kda   The KDA as defined in equation A11 of Etienne 2005. See details section
give.log  Boolean, with default TRUE meaning to return the logarithm of the value
give.like Boolean, with default TRUE meaning to return the likelihood and FALSE meaning to return the probability
start     In function optimal.params(), the start point for the optimization routine \((\theta, m)\).
give      In function optimal.params(), Boolean, with TRUE meaning to return all output of the optimization routine, and default FALSE meaning to return just the point estimate
...       In function optimal.params(), further arguments passed to optim()

Details

Function etienne() is just Etienne’s formula 6:

\[
P[D|\theta, m, J] = \frac{J!}{\prod_{i=1}^{S} n_i \prod_{j=1}^{J} \phi_j!} \theta_S^S \times \sum A=S \left( K(D, A) \frac{(\theta)_J}{(\theta)_A} (I)_J \right)
\]

where \(\log K(D, A)\) is given by function logkda() (qv). It might be useful to know the (trivial) identity for the Pochhammer symbol \([z]_n\) documented in theta.prob.Rd. For convenience, Etienne’s Function optimal.params() uses optim() to return the maximum likelihood estimate for \(\theta\) and \(m\).

Compare function optimal.theta(), which is restricted to no dispersal limitation, ie \(m = 1\).

Argument log.kda is optional: this is the \(K(D, A)\) as defined in equation A11 of Etienne 2005; it is computationally expensive to calculate. If it is supplied, the functions documented here will not have to calculate it from scratch: this can save a considerable amount of time

Author(s)

Robin K. S. Hankin

References


See Also

logkda, optimal.theta
expected.abundance

Examples

data(butterflies)
## Not run: optimal.params(butterflies) # takes too long without PARI/GP

# Now the one from Etienne 2005, supplementary online info:

zoo <- count(c(pigs=1, dogs=1, cats=2, frogs=3, bats=5, slugs=8))
l <- logkda.R(zoo, use.bro=TRUE) # Use logkda() if pari/gp is available
optimal.params(zoo, log.kda=1) # compare his answer of 7.047958 and 0.22635923.

---

expected.abundance  
Expected abundances under the neutral model

Description

Returns a vector of expected abundances of the i-th ranked species under the neutral model

Usage

expected.abundance(J, theta)

Arguments

J  
Size of the ecosystem

theta  
Biodiversity parameter

Value

Returns an object of class count. Species names (capital letters) are assigned by function count().

Note

Function is very slow even for moderate J.

Author(s)

Robin K. S. Hankin

References


See Also

rand.neutral.count
**extractor**  

**Examples**

```r
expected.abundance(I=10,theta=3)
sum(expected.abundance(I=10,theta=3))  #should be 10
```

---

**extractor**  

*Extract rows of a database in count form*

---

**Description**

Extracts rows of a data frame and, if there is one row only, coerces to a count object, preserving the species names.

**Usage**

```r
extractor(x, index)
```

**Arguments**

- `x`  
  A data frame with column headings being species names.

- `index`  
  A vector of indices to extract.

**Details**

If `index` is length one, the numbers are interpreted as species counts, and the output is coerced to a count object.

**Author(s)**

Robin K. S. Hankin

**Examples**

```r
data(saunders)
plot(extractor(saunders.exposed,1))
```
fisher

Various functionality to implement Fisher's logseries

Description

Various functions connected to Fisher's logseries including creation of synthetic datasets and estimation of Fisher's alpha

Usage

fishers.alpha(N, S, give=FALSE)
fisher.ecosystem(N, S, nmax, alpha=NULL, c=0)

Arguments

N Size of the ecosystem. In the case of fisher.ecosystem(), the expected size of the ecosystem
S Number of species in ecosystem
alpha In function fisher.ecosystem(), Fisher's $\alpha$. If not supplied, it will be calculated from $N$ and $S$.
give In function fishers.alpha(), Boolean variable with default FALSE meaning to return alpha, and TRUE meaning to return a list containing $x$ and alpha.
nmax In function fisher.ecosystem(), the maximum number of species abundance classes to consider
c In function fisher.ecosystem(), the rare species advantage term

Details

Function fishers.alpha() solves for $\alpha$ given $N$ and $S$, as per Fisher's table 9, p55. Given $N$ and $S$ (or $\alpha$), function fisher.ecosystem() generates a Fisherian ecosystem with expected size $N$ and expected species count $S$.

Author(s)

Robin K. S. Hankin

References


Examples

fishers.alpha(N=100000, S=100)
#compare the Table value:
100000/10^3.95991
Description

Tree species counts are given in 50 one-hectare sampling plots (species by sample matrix). This only includes trees over 10 cm dbh (diameter at breast height) and species labels (row names) are numeric.

Usage

data(ghats)

Format

Data frame displaying 304 species counts over 50 one-hectare plots.

Source


References


Examples

data(ghats)
# Rank-abundance picture of plot 1 (column 1 in ghat)
plot(extant(count(ghats[,1])))

# histogram of optimal theta across the 50 plots:
hist(apply(ghats,2,optimal.theta),col='gray')
isolate

Randomly select a subset of an ecosystem

Description

Return an ecosystem comprised of individuals randomly sampled from a metacommunity

Usage

\texttt{isolate(a, size = no.of.ind(a), replace = TRUE)}

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Ecosystem data</td>
</tr>
<tr>
<td>size</td>
<td>Number of individuals to sample</td>
</tr>
<tr>
<td>replace</td>
<td>Boolean, with default \texttt{TRUE} meaning to sample individuals from the metacommunity with replacement and \texttt{FALSE} meaning to sample without replacement. See details section</td>
</tr>
</tbody>
</table>

Details

Setting argument \texttt{replace} to default \texttt{TRUE} is much faster.

The canonical example is given by Leigh et al 1993, in which islands were isolated from the mainland by rising waters. The trees on the islands were held to be a randomly drawn sample from the metacommunity.

Given that the usual usage of this function is to generate a plausible ecosystem under such a scenario, one would have a hard time justifying the use of \texttt{replace=TRUE} as it allows (for example) a singleton metacommunity species to have multiple representatives in the returned ecosystem.

However, for large metacommunities and small subsamples, the distinction between \texttt{replace=TRUE} and \texttt{replace=FALSE} is small.

Value

Returns a count object

Note

If \texttt{replace=FALSE}, the returned count object includes extinct species. Use \texttt{extant(isolate(\ldots))} to return only extant species

Author(s)

Robin K. S. Hankin
logkda

References


Examples

```r
a <- rand.neutral(1000,10)
no.of.spp(a)
no.of.spp(isolate(a))
```

---

### Description

Calculates Etienne’s $K(D,A)$ using a variety of different methods

### Usage

```r
logkda.R(a, use.brob=TRUE)
logkda.all(a)
logkda.pari(a, numerical=TRUE, gp_binary = "gp")
logkda.polyn(a)
logkda(a, method="pari", ...)
logkda_pari_unix(a, numerical, pari_string, gp_binary)
logkda_pari_windows(a, numerical, pari_string)
```

### Arguments

- `a` Count object
- `use.brob` In function `logkda.R()`, Boolean, with default TRUE meaning to use Brobdingnagian numbers for the calculation. This is slower but allows one to analyze larger datasets
- `numerical` Boolean, with default TRUE meaning to coerce to a numerical vector (thereby losing precision), and FALSE meaning to return the string produced by pari/gp
- `method` In function `logkda()`, a string specifying which method to use. Takes “R”, “all”, or “pari”
- `pari_string,gp_binary` configuration variables (not intended to be changed by the user)
- `...` In function `logkda()`, further arguments which are passed to the other functions
Details

The user should use function `logkda()`, which is a wrapper for the other functions. Note that the default method, `pari`, requires the pari/gp system to be installed. This is the preferred option because it is much faster than the other methods.

Functions `logkda.R()` and `logkda.pari()` calculate \( K(D, A) \) using the method appearing in Etienne (2005), supplementary online material; they use \( R \) and `pari/gp` respectively. Function `logkda.a11` is a direct implementation of formula A11 in Etienne (2005). The formula is

\[
K(D, A) = \sum_{\{a_1, \ldots, a_S\} | \sum a_i = A} \prod_{i=1}^{S} \frac{\pi(n_i, a_i) \pi(a_i, 1)}{\pi(n_i, 1)}
\]

where \( \pi(n_i, a_i) \) are Stirling numbers of the first kind (see `logs1`).

Function `logkda.pari()` dispatches to either `logkda_pari_unix()` or `logkda_pari_windows()` but the windows function is not guaranteed to work.

Note

If method takes its default value of “pari”, and `pari/gp` is not installed (the test is `gp --version`), then the method is changed to `R` and a warning given.

Function `logkda.a11()` is included because the computational method is a direct transcription of formula A11; it is very slow.

Function `logkda.pari()` is a wrapper for `logkda_pari.windows()` or `logkda_pari_unix()`. It uses “if(R.Version()$os == 'windows')” to check for windows operating systems.

It would be nice to use `gp2c` (rather than `gp`) but I can’t make the “-g” flag work properly; and I had to hack `gp2c-run` to make it call `gp` with the `-q` flag

Author(s)

Robin K. S. Hankin; `logkda()` is an \( R \) transliteration of `pari/gp` code appearing in Etienne 2005 (supplementary online material) due to Chave.

Function `logkda.polyn()` provided by Francois Munoz.

Function `.logkda.pari.windows()` provided by Andrea Manica and Francois Munoz.

References


See Also

`etienne`, `logS1`
Examples

a <- count(c(dogs=7,pigs=3,crabs=1,hogs=1,slugs=1))

## Not run: logkda(a)

logkda.R(a)
logkda.R(a, use.brob=FALSE)
logkda.all(a)
# All four should be the same up to numerical errors

---

logS1  

*logarithms of Stirling numbers of the first kind*

Description

Natural logarithms of Stirling numbers of the first kind, used by function `logkda.all()` (dataset `logS1`) and function `logkda.polyn()` (dataset `logS1vect`).

Usage

logS1

Format

Dataset `logS1` is a 100-by-100 matrix of logs of Stirling numbers of the first kind; `logS1vect` is a vector of length 499500

Source

Calculated by Maple

See Also

`etienne`

Examples

exp(logS1[1:5,1:5])
Ecosystem diagnostics

Description
Ecosystem diagnostics such as species count, individual count, number of singletons, etc

Usage

\begin{verbatim}
no.of.ind(x)
no.of.spp(x, include.extinct=FALSE)
no.of.singletons(x)
no.of.extinct(x)
maximal.abundance(x)
singletons(x)
extinct(x)
extant(x)
\end{verbatim}

Arguments

- \textbf{x} Ecosystem vector; is coerced to class count
- \textbf{include.extinct} In function \texttt{no.of.spp()}, Boolean argument with TRUE meaning to include extinct species (ie species with an abundance of zero), and default FALSE meaning to return the number of extant species

Details

- Function \texttt{no.of.spp()} returns the number of species in an ecosystem object, treating extinct species in line with argument \texttt{include.extinct}
- Function \texttt{no.of.ind()} returns the number of individuals
- Function \texttt{no.of.singletons()} returns the number of singletons
- Function \texttt{no.of.extinct()} returns the number of extinct species
- Function \texttt{maximal.abundance()} returns the abundance of the most abundant species
- Function \texttt{singletons()} returns a count object containing only the singletons: each abundance is one
- Function \texttt{extinct()} returns a count object containing only the extinct species: each abundance is zero
- Function \texttt{extant()} returns a count object containing only the extant species: each abundance is greater than zero

Note

It is sometimes useful to include species with an abundance of zero when, for example, taking a single row of the Saunders dataframe.

The default for \texttt{include.extinct} is FALSE because this is required for (eg) \texttt{optimal.theta()}
optimal.params.gst

Author(s)
Robin K. S. Hankin

References

Examples

data(butterflies)
no.of.spp(butterflies)
no.of.ind(butterflies)

jj1 <- count(c(dogs=7,pigs=3,crabs=1,slugs=1))
jj2 <- count(c(squid=0,octopus=0,nautilus=0))
jj3 <- count(c(bugs=3,rats=1,crabs=0,fish=0))

extinct(jj1 + jj2) # cats 'n' fish
extinct(jj3) # bugs and rats
extant(jj3)

singletons(jj1) # empty
singletons(jj2) # crabs, rats and slugs

Usage

optimal.params.gst(D, exact = TRUE, ci = FALSE, cint = c(0.025, 0.975), nbres = 100)
GST.k(D, exact = TRUE)
I.k(D, exact = TRUE)
Arguments

- **d**: A data table including species counts in a network of community samples (columns)
- **exact**: If `TRUE`, exact similarity statistics are calculated (sampling without replacement) while, if `false`, approximate statistics (sampling with replacement) are considered (see Munoz et al 2008 for further statistical discussion)
- **ci**: Specifies whether bootstraps confidence intervals of immigration estimates are to be calculated
- **cint**: Bounds of the confidence interval, if `ci = TRUE`
- **nbres**: Number of rounds of the bootstrap procedure for confidence interval calculation, if `ci = T`

Value

- **gst**: A vector of 0 to 1 GST(k) numbers (specific output of `GST.k`)
- **nk**: Number of individuals within samples (length = number of samples)
- **distrib**: Species counts of the merged dataset (output of `GST.k` and `I.k`)
- **I**: Immigration estimates (output of `I.k` and `optimal.params.gst`)
- **m**: Corresponding immigration rates (output of `I.k` and `optimal.params.gst`). Specific outputs of `optimal.params.gst` when `ci = T` (bootstrap procedure)
- **Ici**: Confidence interval of I(k)
- **mci**: Confidence interval of m(k)
- **Iboot**: Table of bootstrapped values of I(k)
- **mboot**: Table of bootstrapped values of m(k)

Author(s)

Francois Munoz

References


See Also

- `optimal.params.optimal.params.sloss`

Examples

```r
data(ghats)
optimal.params.gst(ghats)
```
optimal.params.sloss

Estimation of neutral community parameters using a two-stage maximum-likelihood procedure

Description

Function `optimal.params.sloss()` returns maximum likelihood estimates of \( \theta \) and \( m(k) \) using numerical optimization.

It differs from `untb`'s `optimal.params()` function as it applies to a network of smaller community samples \( k \) instead of to a single large community sample.

Although there is a single, common \( \theta \) for all communities, immigration estimates are provided for each local community \( k \), sharing a same biogeographical background.

Usage

```
optimal.params.sloss(D, nbres = 100, ci = FALSE, cint = c(0.025, 0.975))
```

Arguments

- **D**: Species counts over a network of community samples (species by sample table)
- **nbres**: Number of resampling rounds for \( \theta \) estimation
- **ci**: Specifies whether bootstraps confidence intervals should be provided for estimates
- **cint**: Bounds of confidence intervals, if \( ci = T \)

Value

- **\( \theta \)**: Mean \( \theta \) estimate
- **I**: The vector of estimated immigration numbers \( I(k) \)

Output of the bootstrap procedure, if \( ci = T \):

- **thetaci**: Confidence interval for \( \theta \)
- **msampleci**: Confidence intervals for \( m(k) \)
- **thetasamp**: \( \theta \) estimates provided by the resampling procedure
- **Iboot**: Bootstrapped values of \( I(k) \)
- **mboot**: Bootstrapped values of \( m(k) \)

Note

The function returns unhelpful output when run with the `caruso` dataset as in `optimal.params.sloss(caruso)`. The reason for this behaviour is unknown.

Author(s)

Francois Munoz
References

See Also
optimal.params, optimal.params.gst

Examples
data(ghats)
optimal.params.sloss(ghats)

optimal.prob returns an estimate of the fundamental biodiversity number

Description
Returns a maximum likelihood estimate for the fundamental biodiversity number $\theta$ (function optimal.theta()) or the probability of mutation (function optimal.prob()) and optionally return information about the likely error

Usage
optimal.prob(x, interval=NULL, N=NULL, like=NULL, ...)
optimal.theta(x, interval=NULL, N=NULL, like=NULL, ...)

Arguments
x Ecosystem vector or species count table
interval Bracketing interval for probability of mutation to be passed to the optimization routine (here optimize()). Default of NULL means to use a wide interval. Note that this argument is interpreted as an interval of $\theta$ for both optimal.prob() and optimal.theta().
N Integer; the number of parametric resampled estimates to give. Default of NULL means to return just the maximum likelihood estimate
like Units of likelihood to calculate credible interval. Edwards recommends using 2
... Further arguments passed to optimize()

Note
The fundamental biodiversity parameter $\theta$ is $2\nu J$, where $\nu$ is the probability of mutation (ie, as estimated by optimal.prob()), and $J$ is the size of the ecosystem.

For the general case of dispersal limitation, see functions etienne() and optimal.params().
**phi**

**Author(s)**
Robin K. S. Hankin

**See Also**

- etienne
- optimal.params.sloss
- optimal.params.gst

**Examples**

```r
data(butterflies)
optimal.prob(butterflies)
optimal.theta(butterflies)
```

---

### Description

Hubbell’s phi: counts of species abundances

### Usage

```r
phi(x, addnames=TRUE)
unphi(freq, string="spp")
```

### Arguments

- `x` Ecosystem vector; is coerced to class count
- `addnames` Boolean with default TRUE meaning to set the name of the ith element to the species with abundance i if unique. Set to FALSE to suppress this, which is useful if the species names are long
- `freq` Frequency data (eg as returned by `phi()`)
- `string` Character; species name to prepend (using NULL can be confusing)

### Details

Function `phi()` coerces its argument to a count object and by default returns a named vector whose ith element is the number of species with i individuals. The name of the ith element is the species with abundance i if unique and empty otherwise. Function `phi()` is used by `theta.prob()`.

Function `unphi()` does the reverse: given the output of `phi()`, it returns a corresponding count object. Note that species names are lost.

### Note

The code for setting the names is a dog’s breakfast
Author(s)
Robin K. S. Hankin

References

See Also
preston

Examples

jj <- c(rep("oak",5),rep("ash",2),rep("elm",3),"pine","tea","yew")
a <- as.count(jj)

phi(a)
unphi(phi(a))  #should match 'a' except for species names (which are lost)

data(butterflies)
phi(butterflies,add=FALSE)

summary(unphi(phi(butterflies))))  #should match 'summary(butterflies)'

---

plot.count  Abundance curves

Description
Plot the ranked abundance curve

Usage

## S3 method for class 'count'
plot(x, uncertainty = FALSE, expectation = FALSE, theta = NULL, n = 10, ...)
## S3 method for class 'census'
plot(x, uncertainty = FALSE, expectation = FALSE, theta = NULL, n = 10, ...)

Arguments

x  Ecosystem object, coerced to class count
uncertainty  Boolean, with TRUE meaning to show bootstrapped estimates for the species diversity curve, and default FALSE meaning to omit this
expectation  Boolean, with TRUE meaning to plot expected abundances, and default FALSE meaning not to plot them. **Warning** this option takes a looong time to run, even for moderate values of \( J \)
theta Fundamenal biodiversity number used if argument uncertainty or expectation are TRUE. Default value of NULL means to use the maximum likelihood estimate returned by function optimal.theta()

n Number of bootstrapped estimates to plot

... Extra parameters passed to untb()

Details

Plots a ranked abundance curve, optionally with parametrically resampled datasets showing the uncertainties

Note

If using expectation, it’s usually necessary to set ylim and possibly xlim manually.

Author(s)

Robin K. S. Hankin

Examples

data(copepod)
plot(copepod)

data(butterflies)
plot(butterflies,uncertainty=TRUE)

x <- count(c(pigs=1, dogs=1, cats=2, frogs=3, bats=5, slugs=8))
plot(x,expectation=TRUE,ylim=c(0.5,10))
Arguments

- x: Ecosystem vector that is coerced to class count, or a matrix whose rows are species counts.
- n: An integer specifying the number of species abundance classes to use, with default NULL meaning to use \(1 + \log_2(J)\). Must be greater than 1 if specified. If x is a vector, NULL is not acceptable as the program does not try to guess what is required.
- original: Boolean, with default FALSE meaning to use the nonoverlapping technique discussed below, and TRUE meaning to use Preston’s original formulation.

Details

The Preston diagram is a table showing the number of species having abundances in specified abundance classes. Consider the following Preston diagram, created with original = FALSE:

<table>
<thead>
<tr>
<th>number of species</th>
<th>1</th>
<th>2</th>
<th>3-4</th>
<th>5-8</th>
<th>9-16</th>
<th>17-32</th>
<th>33-64</th>
<th>65-Inf</th>
</tr>
</thead>
<tbody>
<tr>
<td>species</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

This shows that there are 10 species with abundance 1 (that is, singletons); 5 species with abundance 2; 7 species with abundance 3-4; 5 species with abundance 5-8, and so on. This method is used by Hubbell (2001), and Chisholm and Burgman (2004).

Setting argument original to TRUE means to follow Preston (1948) and count any species with an abundance on the boundary between two adjacent abundance classes as being split 50-50 between the classes. Thus the fourth class would be \(\phi_4/2 + \phi_5 + \phi_6 + \phi_7 + \phi_8/2\) where \(\phi_i\) is the number of species with abundance \(i\) (given by \(\phi(x)\)).

Value

Function preston() returns an object of class “preston”.

Author(s)

Robin K. S. Hankin

References


See Also

phi
print.preston

Examples

```r
preston(unb(start=rep(1,100), prob=0.01, gens=1000, keep=FALSE))

data(butterflies)
preston(butterflies)
preston(butterflies,original=TRUE)

data(copepod)
preston(copepod)
```

print.preston

Print and plot objects of class Preston

Description

Print and plot objects of class Preston

Usage

```r
## S3 method for class 'preston'
print(x, ...)
## S3 method for class 'preston'
plot(x, ...)
```

Arguments

- `x` Object of class “preston"
- `...` further arguments passed to `print()` after class reset

Note

Intended to work with the output of function `preston()`.
See the vignette for how to annotate a Preston plot.

Author(s)

Robin K. S. Hankin

See Also

`preston`

Examples

```r
data(butterflies)
print(preston(butterflies))
```
print.summary.count  
*Print method for summary objects*

**Description**

Print method for summary objects

**Usage**

```r
## S3 method for class 'summary.count'
print(x, ...)
```

**Arguments**

- `x`  
  Object of class “summary.count”
- `...`  
  extra arguments, currently ignored

**Author(s)**

Robin K. S. Hankin

**Examples**

```r
data(butterflies)
summary(butterflies)
```

---

**rand.neutral**  
*Random neutral ecosystem*

**Description**

Given the size of the metacommunity $J$, and the fundamental biodiversity number $\theta$, generate an object of class `count` using a stochastic mechanism consistent with the neutral theory.

**Usage**

```r
rand.neutral(J, theta=NULL, prob.of.mutate=NULL, string = NULL, pad = FALSE)
```
Arguments

- **J**: Size of metacommunity
- **theta**: Fundamental biodiversity number \( \theta \). User must supply exactly one of \( \theta \) and **prob.of.mutate**.
- **prob.of.mutate**: Probability of mutation \( \nu : \theta = 2J\nu \).
- **string**: String to add to species names. By default (ie string being NULL), species are named “1”, “2”, … Argument string supplies a prefix for these species names; a good one to use is “spp.”. This argument is useful because printing a count object can be confusing if the species names are all integers.
- **pad**: Boolean, with default FALSE meaning to return a count object having only extant species, and TRUE meaning to pad the count with extinct species to \( J \) species. Use this when a vector of length \( J \) is required consistently (see examples section).

Details


Note

If pad is TRUE, and you set string to “extinct”, things will break.

Author(s)

Robin K. S. Hankin

References


See Also

untb

Examples

```r
count <- rand.neutral(1000, 9)
count <- rand.neutral(1000, 9, string="spp.")
data(butterflies)
count <- rand.neutral(no.of.ind(butterflies), optimal.theta(butterflies), string="spp.")
```

# what is the distribution of abundance of the second ranked species if # J=10, theta=0.7? plot(table(replicate(100, rand.neutral(10,theta=0.7,pad=TRUE)[2])))
sahfos  

*Biodiversity dataset provided by SAHFOS*

Description  
Species counts in the North Atlantic

Usage  
data(sahfos)

References  

Examples  
data(sahfos)  
preston(sahfos)

saunders  

*Dataset due to Saunders*

Description  
A dataframe showing species inventories for a kelp holdfast (saunders) including a Boolean flag indicating whether the holdfast was in a sheltered or exposed location.  
Also two data frames, one for the 20 exposed holdfasts (saunders.exposed) and one for the 20 sheltered holdfasts (saunders.sheltered).  
Also three count objects, giving counts for all organisms (saunders.tot), all those from exposed locations (saunders.exposed.tot), and all those from sheltered locations only (saunders.sheltered.tot).

Usage  
data(saunders)

Format  
Dataset saunders is a dataframe with 40 observations on 177 variables. Each row corresponds to a holdfast. The first column is Boolean, indicating whether or not that holdfast was exposed (TRUE) or sheltered (FALSE). The other columns show species abundances for each of 176 species.  
Summary datasets saunders.sheltered.tot, saunders.exposed.tot, and saunders.tot are objects of class count that are the species abundance for sheltered holdfasts, exposed holdfasts, and the entire dataset.
The user will probably be most interested in `saunders.sheltered` and `saunders.exposed`, which are the transpose of the appropriate rows of `saunders`. Thus these dataframes have 176 rows, one per species and 20 rows, one per holdfast.

Details

Kelp are large seaweeds classified in kingdom Chromista. Kelp grows in shallow oceans in kelp forests.

The *holdfast* is a root-like structure that anchors the kelp to the ocean floor. Fauna inhabiting kelp holdfasts, being “incredibly diverse” (Anderson et al 2005), are often used as indicators of environmental change.

The data was collected in New Zealand, from eight sites along the Leigh coastline from north of Leigh Harbour down to the southern end of Kawau Island (a stretch of roughly 20 km). Four sites were wave-exposed, four were sheltered (although two of the latter were arguably quite tidally-dominated). Each site had a spatial extent of roughly one hectare. They were collected from 5 - 10 November, 2003.

The `saunders` dataset must be arranged as it is because if it were transposed, the first row would be the (nonsensical) observation c(T,T,…,T,F,…,F).

Note

It is not entirely obvious how to derive the summary datasets from the `saunders` dataframe. Use function `extractor()` for this.

Source

Data supplied by Justine Saunders

References


See Also

`extractor`

Examples

```r
data(saunders)
plot(saunders.sheltered.tot, uncertainty=TRUE, n=1)
preston(saunders.tot)
optimal.params.sloss(saunders.exposed)
```
simpson

Simpson’s diversity index

Description

Simpson’s diversity index

Usage

simpson(x, with.replacement=FALSE)

Arguments

x Ecosystem vector; coerced to class count

with.replacement

Boolean, with default FALSE meaning to sample without replacement; see details section

Details

Returns the Simpson index \( D \): the probability that two randomly sampled individuals belong to different species.

There is some confusion as to the precise definition: some authors specify that the two individuals are necessarily distinct (ie sampling without replacement), and some do not.

Simpson (1949) assumed sampling without replacement and gave

\[
1 - \sum_{i=1}^{S} \frac{n_i (n_i - 1)}{J(J - 1)}
\]

in our notation.

He and Hu (2005) assumed sampling with replacement:

\[
1 - \frac{\sum_{i=1}^{S} n_i^2}{J^2}.
\]

The difference is largely academic but is most pronounced when many species occur with low counts (ie close to 1).

Author(s)

Robin K. S. Hankin

References

species.count

See Also

preston

Examples

data(butterflies)

D <- simpson(butterflies)
theta <- optimal.prob(butterflies)*2*no.of.ind(butterflies)

# compare theta with D/(1-D) (should be roughly equal; see He & Hu 2005):
theta
D/(1-D)

# Second argument pedantic in practice.

# Mostly, the difference is small:
simpson(butterflies,FALSE) - simpson(butterflies,TRUE)

# Most extreme example:
x <- count(c(1,1))
simpson(x,TRUE)
simpson(x,FALSE)

species.count  Ecosystem diagnostics for output of untb()  

Description

Provides ecosystem diagnostics of species count datasets (species counts and species tables), useful for the output of untb()

Usage

species.count(x)
species.table(x)

Arguments

x  An integer matrix whose rows are integers representing the individuals’ species
Details

These functions takes a matrix argument, which is interpreted as the output of `untb(..., keep=TRUE)`. Function `species.count()` returns the total number of species present in each row (ie at each timestep). Function `species.table()` returns a matrix $M$ where $M[i,j]$ column of the matrix is the abundance of species $j$ at time $i$.

Author(s)

Robin K. S. Hankin

See Also

`preston`

Examples

```r
a <- untb(start=rep(1,50), prob=0.01, gens=2000, keep=TRUE)

plot(species.count(a), type="b")
matplot(species.table(a), type="l", lty=1)

jj <- a[2000,]
print(jj)
as.count(jj)
```

---

<table>
<thead>
<tr>
<th>spitale</th>
<th>Counts of diatom species in springs of the Adamello-Brenta Nature Park</th>
</tr>
</thead>
</table>

Description

A dataset due to Spitale and Cantonati comprising abundances of different species of diatoms

Usage

`data(spitale)`

Format

A count object

Source

Data kindly provided by Daniel Spitale
**summary.count**

**References**


**Examples**

```r
data(spitale)
summary(spitale)
```

**Description**

Summary methods for count and census objects

**Usage**

```r
## S3 method for class 'count'
summary(object, ...)
## S3 method for class 'census'
summary(object, ...)
```

**Arguments**

- `object` Ecosystem object coerced to class count
- `...` Further arguments, currently ignored

**Details**

Prints a summary of an ecosystem object.

**Author(s)**

Robin K. S. Hankin

**See Also**

`phi`

**Examples**

```r
data(ostracod)
summary(ostracod)
```
Describes the posterior probability and likelihood for \( \theta \), given a count object

Usage

\[
\text{theta.prob}(\theta, \text{x} = \text{NULL}, \text{give.log} = \text{TRUE})
\]

\[
\text{theta.likelihood}(\theta, \text{x} = \text{NULL}, \text{S} = \text{NULL}, \text{J} = \text{NULL}, \text{give.log} = \text{TRUE})
\]

Arguments

- \( \theta \): biodiversity parameter
- \( \text{x} \): object of class count or census
- \( \text{give.log} \): Boolean, with FALSE meaning to return the value, and default TRUE meaning to return the (natural) logarithm of the value
- \( \text{S}, \text{J} \): In function theta.likelihood\(), the number of individuals (\( J \)) and number of species (\( S \)) in the ecosystem, if \( \text{x} \) is not supplied. These arguments are provided so that \( \text{x} \) need not be specified if \( \text{S} \) and \( \text{J} \) are known.

Details

The formula was originally given by Ewens (1972) and is shown on page 122 of Hubbell (2001):

\[
\frac{1^{\phi_1} 2^{\phi_2} \cdots J^{\phi_J} \prod_{k=1}^{J} (\theta + k - 1)}{\prod_{k=1}^{J} (\theta + k - 1)}. 
\]

The likelihood is thus given by

\[
\frac{\theta^S}{\prod_{k=1}^{J} (\theta + k - 1)}. 
\]

Etienne observes that the denominator is equivalent to a Pochhammer symbol \((\theta)_J\), so is thus readily evaluated as \( \Gamma(\theta + J)/\Gamma(\theta) \) (Abramowitz and Stegun 1965, equation 6.1.22).

Note

If estimating \( \theta \), use theta.likelihood\() rather than theta.probability\() because the former function generally executes much faster: the latter calculates a factor that is independent of \( \theta \).

The likelihood function \( L(\theta) \) is any function of \( \theta \) proportional, for fixed observation \( z \), to the probability density \( f(z, \theta) \). There is thus a slight notational inaccuracy in speaking of “the” likelihood function which is defined only up to a multiplicative constant. Note also that the “support” function is usually defined as a likelihood function with maximum value 1 (at the maximum likelihood estimator for \( \theta \)). This is not easy to determine analytically for \( J > 5 \).
Note that $S$ is a sufficient statistic for $\theta$.

Function `theta.prob()` does not give a PDF for $\theta$ (so, for example, integrating over the real line does not give unity). The PDF is over partitions of $J$; an example is given below.

Function `theta.prob()` requires a count object (as opposed to `theta.likelihood()`, for which $J$ and $S$ are sufficient) because it needs to call `phi()`.

**Author(s)**

Robin K. S. Hankin

**References**


**See Also**

`phi`, `optimal.prob`

**Examples**

\[
\text{theta.prob}(1, \text{rand.neutral}(15, \text{theta}=2))
\]

\[
gg <- \text{as.count(c(rep("a", 10), rep("b", 3), letters[5:9]))}
\]
\[
\text{theta.likelihood}(\text{theta}=2, gg)
\]
\[
\text{optimize}(f=\text{theta.likelihood}, \text{interval}=c(0, 100), \text{maximum}=\text{TRUE}, x=gg)
\]

## An example showing that `theta.prob()` is indeed a PMF:

\[
a <- \text{count(c(dogs=3, pigs=3, hogs=2, crabs=1, bugs=1, bats=1))}
\]
\[
x <- \text{partitions::parts(no.of.ind(a))}
\]
\[
f <- \text{function(x)}(\text{theta.prob}(\text{theta}=1.123, \text{extant(count(x)}, \text{give.log}=\text{FALSE}))
\]
\[
\text{sum(apply(x, 2, f))} \quad \# \text{ should be one exactly}.
\]

**untb**

*Ecological drift simulation under the Unified Neutral Theory of Biodiversity*

**Description**

Simulates ecological drift under the UNTB. Function `untb()` carries out the simulation; function `select()` carries out a single generational step.
Usage

untb(start, prob=0, D=1, gens=150, keep=FALSE, meta=NULL)
select(a, D=length(a), prob=0, meta=NULL)
select.mutate(a, D=length(a), prob.of.mutate=0)
select.immigrate(a, D=length(a), prob.of.immigrate=0, meta)

Arguments

a, start Starting ecosystem; coerced to class census. Usually, pass an object of class count; see examples. To start with a monoculture of size 10, use start=rep(1,10) and to use start=1:10.
prob, prob.of.immigrate, prob.of.mutate Probability of “new” organism not being a descendent of an existing individual
D Number of organisms that die in each timestep
gens Number of generations to simulate
keep In function untb() Boolean with default FALSE meaning to return the system at the end of the simulation and TRUE meaning to return a matrix whose rows are the ecosystem at successive times
meta In function untb(), the metacommunity; coerced to a count object. Default of NULL means to use a “greedy” system in which every mutation gives rise to a new, previously unencountered species. This would correspond to an infinitely large, infinitely diverse, Hubbellian ecosystem (which is not too ridiculous an assumption for a small island near a large diverse mainland). In function select.immigrate(), a simplified representation of a metacommunity.

Details

Functions select.immigrate() and select.mutate() are not really intended for the end user; they use computationally efficient (and opaque) integer arithmetic.

Author(s)

Robin K. S. Hankin

References


Examples

data(butterflies)
untb(start=butterflies, prob=0, gens=100)

a <- untb(start=1:10,prob=0.005, gens=1000,keep=TRUE)
plot(species.count(a),type="b")
matplot(species.table(a),type="1",lty=1)
Various functions from Vallade and Houchmandzadeh (2003), dealing with analytical solutions of a neutral model of biodiversity

Usage

vallade.eqn5(JM, theta, k)
vallade.eqn7(JM, theta)
vallade.eqn12(J, omega, m, n)
vallade.eqn14(J, theta, m, n)
vallade.eqn16(J, theta, mu)
vallade.eqn17(mu, theta, omega, give=FALSE)

Arguments

\(J, JM\) Size of the community and metacommunity respectively
theta Biodiversity number \(\theta = (J_M - 1)\nu/(1 - \nu)\) as discussed in equation 6
k Abundance
omega Relative abundance \(\omega = k/J_M\)
m Immigration probability
mu Scaled immigration probability \(\mu = (J - 1)m/(1 - m)\)
give In function vallade.eqn17(), Boolean with default FALSE meaning to return the numerical value of the integral and TRUE meaning to return the entire output of integrate() including the error estimates

Details

Notation follows Vallade and Houchmandzadeh (2003) exactly.

Note

Function vallade.eqn16() requires the polynom library, which is not loaded by default. It will not run for \(J > 50\) due to some stack overflow error.

Function vallade.eqn5() is identical to function alonso.eqn6()

Author(s)

Robin K. S. Hankin
References


Examples

# A nice check:
JM <- 100
k <- 1:JM
sum(k*vallade.eqn5(JM, theta=5, k)) # should be JM=100 exactly.

# Now, a replication of Figure 3:
omega <- seq(from=0.01, to=0.99, len=100)
f <- function(omega, mu){
vallade.eqn17(mu, theta=5, omega=omega)
}
plot(omega,
omega*5,type="n",xlim=c(0,1),ylim=c(0,5),

xlab=expression(omega),
ylab=expression(omega*g[C](omega)),
main="Figure 3 of Vallade and Houchmandzadeh")
points(omega,omega*sapply(omega,f,mu=0.5),type="l")
points(omega,omega*sapply(omega,f,mu=1),type="l")
points(omega,omega*sapply(omega,f,mu=2),type="l")
points(omega,omega*sapply(omega,f,mu=4),type="l")
points(omega,omega*sapply(omega,f,mu=8),type="l")
points(omega,omega*sapply(omega,f,mu=16),type="l")
points(omega,omega*sapply(omega,f,mu=Inf),type="l")

# Now a discrete version of Figure 3 using equation 14:
j <- 100
omega <- (1:j)/j

f <- function(n,mu){
m <- mu/(J-1+mu)
vallade.eqn14(J=j, theta=5, m=m, n=n)
}
plot(omega,omega*0.03,type="n",main="Discrete version of Figure 3 using
eqn 14")
points(omega,omega*sapply(1:j,f,mu=16))
points(omega,omega*sapply(1:j,f,mu=8))
points(omega,omega*sapply(1:j,f,mu=4))
points(omega,omega*sapply(1:j,f,mu=2))
points(omega,omega*sapply(1:j,f,mu=1))
points(omega,omega*sapply(1:j,f,mu=0.5))
volkov

Expected frequency of species

Description

Given a community size, biodiversity parameter $\theta$, and an immigration rate $m$, returns the expected frequency of species with $n$ individuals, for $0 < n \leq J$.

Usage

```
volkov(J, params, bins = FALSE, give = FALSE)
```

Arguments

- `J` Size of community
- `params` A two-element vector with first element interpreted as theta, the Fundamental biodiversity parameter and the second, $m$, interpreted as the probability of immigration. This argument will accept the output of `optimal.params()`
- `bins` Boolean, with default `FALSE` meaning to return the expected number of species with $1, 2, \ldots, J$ individuals, and `FALSE` meaning to return the binned total, using a Preston-like binning system as used in `preston()`
- `give` Boolean, with `TRUE` meaning to return all the output of `integrate()`, and default `FALSE` meaning to return just the value of the integral

Value

Returns an object of class “phi”.

Note

The method used is slightly inefficient: the terms to the left of the integral sign [in Volkov’s equation 7] are integrated and this is, strictly, unnecessary as it is not a function of $y$. However, taking advantage of this fact results in messy code.

Author(s)

Robin K. S. Hankin

References


See Also

`phi`, `preston`
Examples

```r
## Not run:
volkov(J=21457,c(theta=47.226, m=0.1)) # Example in figure 1

## End(Not run)
volkov(J=20, params=c(theta=1, m=0.4))

data(butterflies)
r <- plot(preston(butterflies, n=9, orig=TRUE))

## Not run: jj <- optimal.params(butterflies) # needs PARI/GP
jj <- c(9.9980936124759, 0.991791987473506)
points(r, volkov(no.of.ind(butterflies), jj, bins=TRUE), type="b")
```

---

**zsm**

*Zero sum multinomial distribution as derived by McKane*

**Description**

The Zero sum multinomial distribution of species abundances as derived by McKane 2004.

**Usage**

`zsm(J, P, m)`

**Arguments**

- `J` Size of local community
- `P` Abundance in metacommunity
- `m` Probability of immigration

**Value**

Returns a vector of size `J` showing the probability of the stationary abundance being `1,...,J`.

**Note**

The function uses `lgamma()` to avoid numerical overflow

**Author(s)**

Robin K. S. Hankin
References


Examples

```r
sum(zsm(164,0.1,0.5))  # should be 1
```

# McKane et al 2004: figure 1.
```r
layout(matrix(1:4,2,2))
par(mai=0.2+rep(0,4))
plot(1,type="n",log="y",ylim=c(1e-9,1),xlim=c(0,64),xlab="",ylab="Ps(N)",
axies=FALSE,main=expression(J==64))
axis(1,pos=1e-9)
axis(2,pos=0,at=10^c(-0:9))
segments(64,1e-9,64,1)
segments(60,1e-9,64,1e-9)
f <- function(P){points(0:64,zsm(64,P,m=0.05),type="l")
for(i in 1:9){f(i/10)}
f(0.99)
f(0.999)
f(0.01)
f(0.001)
text(87,3.2e-7,adj=0,expression(P==0.999))
text(49,3.2e-7,adj=0,expression(P==0.001))
text(45,0.1,expression(m==0.05))

plot(1,type="n",log="y",ylim=c(1e-5,1),xlim=c(0,64),xlab="",ylab="Ps(N)",
axies=FALSE,main="")
axis(1,pos=1e-5)
axis(2,pos=0,at=10^c(-0.5))
segments(60,1e-5,64,1e-5)
segments(64,1e-5,64,1)
par(xpd=FALSE)
g <- function(m){points(0:64,pmax(zsm(64,P=0.1,m=m),1e-5),type="l")
g(0.0001)
g(0.0005)
g(0.01)
g(0.02)
g(0.05)
g(0.5)
g(0.999)
text(50,0.4,expression(P==0.1))

plot(1,type="n",log="y",ylim=c(1e-9,1),xlim=c(0,1e5),xlab="",ylab="Ps(N)",
axies=FALSE,main=expression(J==10000))
axis(1,pos=1e-9)
axis(2,pos=0)
```
segments(1e5,1e-9,1e5,0.1)

h <- function(P)(points(0:1e5,pmax(zsm(1e5,P=m=0.05),1e-9),type="l"))
for(i in 1:9)(h(i/10))
h(0.01)
h(0.99)
text(75000,0.1,expression(m==0.5))

plot(1,type="n",log="y",ylim=c(1e-40,1),xlim=c(0,1e5),xlab="",ylab="Ps(N)",
axes=FALSE,main="")
axis(1,pos=1e-40)
axis(2,pos=0,at=1/10*c(40,32,24,16,8,0))
segments(1e5,1e-40,1e5,1)

i <- function(m)(points(0:1e5,pmax(zsm(1e5,P=0.1,m=m),1e-40),type="l"))
i(0.001)
i(0.002)
i(0.005)
i(0.01)
i(0.02)
i(0.5)
text(60000,1e-4,expression(P==0.1))
Index

*Topic **datasets**
  bci, 5
  butterflies, 6
  caruso, 6
  copepod, 9
  ghat, 17
  logSt, 21
  sahfo, 34
  saunders, 34
  spitale, 38

*Topic **math**
  +.count, 3
  alonso, 4
  census, 8
  count, 10
  display.unbt, 11
  etienne, 12
  expected.abundance, 14
  extractor, 15
  fisher, 16
  isolate, 18
  logkda, 19
  no.of.ind, 22
  optimal.prob, 26
  phi, 27
  plot.count, 28
  preston, 29
  print.preston, 31
  print.summary.count, 32
  rand.neutral, 32
  simpson, 36
  species.count, 37
  summary.count, 39
  theta.prob, 40
  unbt, 41
  unbt-package, 2
  vallade, 43
  volkov, 45
  zsm, 46

*Topic **optimize**
  optimal.params.gst, 23
  optimal.params.sloss, 25
  +.census(+.count), 3
  +.count, 3
  alonso, 4
  as.census(census), 8
  as.count(count), 10
  BCI (bci), 5
  bci, 5
  butterflies, 6
  butterfly (butterflies), 6
  Caruso (caruso), 6
  caruso, 6
  census, 8, 11
  copepod, 9
  count, 8, 10, 14
  display.unbt, 11
  Etienne (etienne), 12
  etienne, 12, 20, 21, 27
  expected.abundance, 14
  extant (no.of.ind), 22
  extinct (no.of.ind), 22
  extractor, 7, 15, 35
  fisher, 16
  fishers.alpha (fisher), 16
  ghats, 17
  GST.k (optimal.params.gst), 23
  I.k (optimal.params.gst), 23
  is.census (census), 8
  is.count (count), 10
  isolate, 18
  logkda, 13, 19
logkda_pari_unix (logkda), 19
logkda_pari_windows (logkda), 19
logS1, 20, 21
logS1vect (logS1), 21

maximal.abundance (no.of.ind), 22
no.of.extinct (no.of.ind), 22
no.of.ind, 22
no.of.singletons (no.of.ind), 22
no.of.spp (no.of.ind), 22

optimal.params, 24, 26
optimal.params (etienne), 12
optimal.params.gst, 23, 26, 27
optimal.params.sloss, 24, 25, 27
optimal.prob, 26, 41
optimal.theta, 13
optimal.theta (optimal.prob), 26
Oribatid (caruso), 6
oribatid (caruso), 6
ostracod (copepod), 9

phi, 27, 30, 39, 41, 45
plot.census (plot.count), 28
plot.count, 28
plot.preston (print.preston), 31
Preston (preston), 29
preston, 28, 29, 31, 37, 38, 45
print.preston, 31
print.summary.census
  (print.summary.count), 32
print.summary.count, 32

rand.neutral, 14, 32

sahfos, 34
saunders, 34
select (untb), 41
simpson, 36
singleton (no.of.ind), 22
species.count, 37
species.table (species.count), 37
spitale, 38
summary.census (summary.count), 39
summary.count, 39

theta.likelihood (theta.prob), 40
theta.prob, 40

unphi (phi), 27
untb, 33, 41
untb-package, 2
Vallade (vallade), 43
vallade, 43
volkov, 45
zsm, 46