Package ‘GUILDS’

October 12, 2022

Type Package

Title Implementation of Sampling Formulas for the Unified Neutral Model of Biodiversity and Biogeography, with or without Guild Structure

Version 1.4.5

Description A collection of sampling formulas for the unified neutral model of biogeography and biodiversity. Alongside the sampling formulas, it includes methods to perform maximum likelihood optimization of the sampling formulas, methods to generate data given the neutral model, and methods to estimate the expected species abundance distribution. Sampling formulas included in the GUILDS package are the Etienne Sampling Formula (Etienne 2005), the guild sampling formula, where guilds are assumed to differ in dispersal ability (Janzen et al. 2015), and the guilds sampling formula conditioned on guild size (Janzen et al. 2015).

License GPL-2

Imports Rcpp (>= 0.11.0), pracma, nloptr

Suggests testthat, knitr, rmarkdown

LinkingTo Rcpp

NeedsCompilation yes

Encoding UTF-8

URL https://github.com/thijsjanzen/GUILDS

BugReports https://github.com/thijsjanzen/GUILDS/issues

VignetteBuilder knitr

Author Thijs Janzen [aut, cre],
    Bart Haegeman [ctb],
    Franck Jabot [ctb],
    Jerome Chave [ctb]

Maintainer Thijs Janzen <thijsjanzen@gmail.com>

Repository CRAN

Date/Publication 2022-03-24 15:40:08 UTC
GUILDS-package

Package implementing the Guilds sampling formula for the Neutral Theory of Biodiversity

Description

The GUILDS package contains a number of sampling formula’s being the Etienne Sampling Formula (Etienne 2005), the GUILDS sampling formula (Janzen et al. 2014) and the GUILDS sampling formula conditioned on guild Size (Janzen et al. 2015). Furthermore it contains functions to generate data given the guilds model, with or without conditioning on guild size. C++ Code to obtain Sterling numbers of the first kind was adopted from the Tetame program by Jabot et al. (2008).

Updates

Version 1.4 : Cleaner README and Vignettes
Version 1.4 : Extend support to M1 processors where sizeof(long double) < 16
Version 1.4 : Comply with _R_CHECK_LENGTH_0_LOGIC2_
Version 1.3 : GUILDS is now on GitHub: https://github.com/thijsjanzen/GUILDS
Version 1.3 : Wrote code tests to check code integrity, code coverage is >95%
Version 1.3 : Modified maximum likelihood functions to take into account theta_x = theta_y = theta / 2
Version 1.3 : Added a plotting function to plot Preston style plots
Version 1.2.1 : Updated the User manual
Version 1.2 : fixed memory leak issues by adding extra vector access checks
Version 1.2 : fixed memory leak issues by introducing vectors in KDA code
Version 1.2 : renamed logLik to avoid shadowing of the function logLik in the package stats
Version 1.1 : removed malloc header from KDA code
expected.SAD

Details

Package: GUILDS
Type: Package
Version: 1.3
License: GPL 2.0

Author(s)

Thijs Janzen
Maintainer: Thijs Janzen <thijsjanzen@gmail.com>

References


expected.SAD

Calculate the expected species abundance distribution of the standard neutral model, given theta, m and J

Description

This function calculates the expected species abundance distribution of the standard neutral model given theta, m and J, sensu equation 6 from Etienne and Alonso (2005).

Usage

expected.SAD(\theta, m, J)

Arguments

\theta \hspace{0.5cm} \text{Fundamental biodiversity number theta}

m \hspace{0.5cm} \text{migration parameter}

J \hspace{0.5cm} \text{Total number of individuals in the local community}

Value

A vector containing the abundances binned into log2 bins (sensu Preston).
**expected.SAD.Guilds**

Estimate the expected species abundance distribution of both guilds using the guilds model, provided theta, alpha_x, alpha_y and J.

**Description**

This function estimates the expected species abundance distribution of both guilds using the guilds model, provided theta, alpha_x, alpha_y and J. The expected species abundance distribution is approximated by first drawing px from a beta distribution (equation 4 in Janzen et al. 2014). Then, guild sizes are drawn using equation 3 in Janzen et al. 2014. Because the abundance distributions of the two guilds are independent, the distributions can now be obtained using equation 6 in Etienne and Alonso 2005. Because drawing from the beta distribution and equation 3 is inherently stochastic, this function returns the average over a specified number of replicates.

**Usage**

```r
expected.SAD.Guilds(theta, alpha_x, alpha_y, J, n_replicates = 100)
```

**Arguments**

- `theta`: Fundamental biodiversity number theta
- `alpha_x`: Dispersal ability of guild X
- `alpha_y`: Dispersal ability of guild Y
- `J`: Total number of individuals in the local community, e.g. \( J = J_x + J_y \)
- `n_replicates`: Number of replicates to use to estimate the abundance distributions.

**Value**

- `guildX`: Vector containing the mean abundances of species in Guild X, binned into log2 bins
- `guildY`: Vector containing the mean abundances of species in Guild Y, binned into log2 bins
expected.SAD.Guilds.Conditional

Author(s)
Thijs Janzen & Bart Haegeman

References

Examples
SADs <- expected.SAD.Guilds(theta = 42, alpha_x = 0.01, alpha_y = 0.1, J = 1000, n_replicates = 3)
par(mfrow=c(1,2));
barplot(SADs$guildX, names.arg=0:(length(SADs$guildX)-1),
 xlab="Number of individuals (log2)",
 ylab="Number of Species",main="Guild X" )

barplot(SADs$guildY, names.arg=0:(length(SADs$guildY)-1),
 xlab="Number of individuals (log2)",
 ylab="Number of Species",main="Guild Y" )

expected.SAD.Guilds.Conditional

*Estimate the expected species abundance distribution of both guilds using the guilds model, provided theta, alpha_x, alpha_y, conditional on the size of guild X, Jx and the size of guild Y, Jy.*

Description
This function estimates the expected species abundance distribution of both guilds using the guilds model, provided theta, alpha_x, alpha_y and J. The expected species abundance distribution is approximated by first drawing px from equation 9. Because the abundance distributions of the two guilds are independent, the distributions can now be obtained using equation 6 in Etienne and Alonso 2005. Because drawing from the beta distribution and equation 3 is inherently stochastic, this function returns the average over a specified number of replicates.

Usage
expected.SAD.Guilds.Conditional(theta, alpha_x, alpha_y, Jx, Jy, n_replicates = 100)

Arguments
theta Fundamenta biodiversity number theta
alpha_x Dispersal ability of guild X
alpha_y Dispersal ability of guild Y
Jx Total number of individuals in guild X
Jy Total number of individuals in guild Y
n_replicates Number of replicates to use to estimate the abundance distributions.
generate.ESF

**Value**

= guildX Vector containing the mean abundances of species in Guild X, binned into log2 bins

= guildY Vector containing the mean abundances of species in Guild Y, binned into log2 bins

**Author(s)**

Thijs Janzen & Bart Haegeman

**References**


**Examples**

```r
SADs <- expected.SAD.Guilds.Conditional(theta = 42, 
   alpha_x = 0.01, 
   alpha_y = 0.1, 
   Jx = 100, 
   Jy = 200, 
   n_replicates = 3)
par(mfrow=c(1,2))
barplot(SADs$guildX, names.arg=0:(length(SADs$guildX) - 1), 
    xlab = "Number of individuals (log2)", 
    ylab = "Number of Species", main = "Guild X")
barplot(SADs$guildY, names.arg = 0:(length(SADs$guildY) - 1), 
    xlab = "Number of individuals (log2)", 
    ylab = "Number of Species", main = "Guild Y")
```

---

**generate.ESF**  
*Generate community data under the standard neutral model of biodiversity, using the urn scheme as described in Etienne 2005*

**Description**

This function generates community data under the standard neutral model of biodiversity, using the urn scheme as described in Etienne 2005

**Usage**

```r
generate.ESF(theta, I, J)
```
**generate.Guilds**

**Arguments**
- **theta**: Fundamental biodiversity number theta
- **I**: Fundamental dispersal number I
- **J**: total number of individuals in the local community

**Value**
Vector containing the unlabeled species abundances in the local community

**Author(s)**
Thijs Janzen & Bart Haegeman

**References**

**Examples**
```r
generate.Guilds(theta = 42, I = 10, J = 2000)
```

---

Using this function it is possible to generate a community dataset consisting of two separate abundance vectors for each guild, where the data generated adhere to the Guilds model.

**Usage**
```r
generate.Guilds(theta, alpha_x, alpha_y, J)
```

**Arguments**
- **theta**: Fundamental Biodiversity Number theta
- **alpha_x**: Dispersal Ability of Guild X
- **alpha_y**: Dispersal Ability of Guild Y
- **J**: Total number of individuals in the local community (e.g. \(J_X + J_Y\)).

**Value**
- **guildX**: Vector containing the unlabeled abundances of species in Guild X
- **guildY**: Vector containing the unlabeled abundances of species in Guild Y
**Author(s)**

Thijs Janzen

**Examples**

```r
generate.Guilds(theta = 200,
                alpha_x = 0.005,
                alpha_y = 0.001,
                J = 10000)
```

---

**generate.Guilds.Cond**  
*Generate Artificial data under the GUILDS model, conditioned on Guild size*

**Description**

Using this function it is possible to generate a community dataset consisting of two separate abundance vectors for each guild, where the data generated adhere to the Guilds model. Data generated is conditioned on guild size.

**Usage**

```r
generate.Guilds.Cond(theta, alpha_x, alpha_y, JX, JY)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>theta</td>
<td>Fundamental Biodiversity Number theta</td>
</tr>
<tr>
<td>alpha_x</td>
<td>Dispersal Ability of Guild X</td>
</tr>
<tr>
<td>alpha_y</td>
<td>Dispersal Ability of Guild Y</td>
</tr>
<tr>
<td>JX</td>
<td>Total number of individuals in Guild X</td>
</tr>
<tr>
<td>JY</td>
<td>Total number of individuals in Guild Y</td>
</tr>
</tbody>
</table>

**Value**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>guildX</td>
<td>Vector containing the unlabeled abundances of species in Guild X</td>
</tr>
<tr>
<td>guildY</td>
<td>Vector containing the unlabeled abundances of species in Guild Y</td>
</tr>
</tbody>
</table>

**Author(s)**

Thijs Janzen

**Examples**

```r
generate.Guilds.Cond(theta = 200,
                      alpha_x = 0.005,
                      alpha_y = 0.001,
                      JX = 15000,
                      JY = 5000);
```
**logLikelihood.ESF**  

**Likelihood of the Etienne sampling formula**

---

**Description**

This function calculates the likelihood of the Etienne Sampling Formula, provided abundance data and parameter values.

**Usage**

```r
logLikelihood.ESF(theta, m, abund)
```

**Arguments**

- `theta` Parameter value for the fundamental biodiversity number theta
- `m` Parameter value for migration
- `abund` Vector containing abundance data

**Value**

Returns the LogLikelihood

**Author(s)**

Thijs Janzen

**References**


**Examples**

```r
A <- c(1,1,1,3,5,8); # Artificial abundance dataset
LL <- logLikelihood.ESF(theta = 7, m = 0.1, abund = A)
```
logLikelihood.Guilds  

Likelihood of the Guilds sampling formula

Description

This function calculates the likelihood of the guilds model, provided abundance data and parameter values.

Usage

logLikelihood.Guilds(parameters, model, sadx, sady, verbose = TRUE)

Arguments

parameters  
parameters corresponds to a vector of parameter values depending on the provided model:
- model: "D0" parameters = c(theta, alpha)
- model: "D1" parameters = c(theta, alpha X, alpha Y)

model  
The chosen model to calculate the likelihood for, please note that the vector of parameters should contain the corresponding parameters in the right order. The user can pick one of these models:
- "D0"
- "D1"

sadx  
The Species Abundance Distribution of guild X

sady  
The Species Abundance Distribution of guild Y

verbose  
TRUE/FALSE flag, indicates whether intermediate output is shown on screen

Value

returns the LogLikelihood

Author(s)

Thijs Janzen

Examples

```r
exampleData <- generate.Guilds(theta = 200,
                               alpha_x = 0.005,
                               alpha_y = 0.001,
                               J = 1000)
#theta = 200, alpha X = 0.005, alpha Y = 0.001
parametervals <- c(200, 0.005, 0.001)
LL = logLikelihood.Guilds(parametervals,
                          model = "D1",
```
logLikelihood.Guilds.Conditional

Likelihood of the Guilds sampling formula, conditional on guild size

Description

This function calculates the likelihood of the guilds model, conditional on guild size: provided abundance data and parameter values.

Usage

logLikelihood.Guilds.Conditional(parameters, model, sadx, sady, verbose = TRUE)

Arguments

parameters parameters corresponds to a vector of parameter values depending on the provided model:
- model: "D0" parameters = c(theta, alpha)
- model: "D1" parameters = c(theta, alpha X, alpha Y)

model The chosen model to calculate the likelihood for, please note that the vector of parameters should contain the corresponding parameters in the right order. The user can pick one of these models:
- "D0"
- "D1"

sadx The Species Abundance Distribution of guild X

sady The Species Abundance Distribution of guild Y

verbose TRUE/FALSE flag, indicates whether intermediate output is shown on screen

Value

returns the LogLikelihood

Author(s)

Thijs Janzen
Examples

```r
exampleData <- generate.Guilds.Cond(theta = 200,
alpha_x = 0.005,
alpha_y = 0.001,
JX = 1000,
JY = 2000)
#theta = 200, alpha X = 0.005, alpha Y = 0.001
parametervals <- c(200, 0.005, 0.001)
LL = logLikelihood.Guilds.Conditional(parametervals,
   model="D1",
   exampleData$guildX,
   exampleData$guildY,
   verbose=TRUE)
```

Description

This function computes the maximum likelihood estimates of the parameters of the Neutral model, using the Etienne Sampling Formula

Usage

```r
maxLikelihood.ESF(init_vals, abund, verbose = FALSE)
```

Arguments

- `init_vals` A vector of initial starting values, of the format c(theta, m)
- `abund` Vector containing a record of the number of individuals per species
- `verbose` TRUE/FALSE flag, indicates whether intermediate output is shown on screen

Value

The output is a list containing the following:

- `par` a vector containing the parameter values at the maximum likelihood c(theta, m)
- `fvalues` the likelihood at the corresponding parameter values
- `conv` gives a message on convergence of optimization; conv = 0 means convergence

Author(s)

Thijs Janzen
**References**


**Examples**

```r
A <- c(1, 1, 1, 3, 5, 8)
maxLikelihood.ESF( c(7, 0.1), abund = A)
```

**Description**

This function computes the maximum likelihood estimates of the parameters of the guilds model.

**Usage**

```r
maxLikelihood.Guilds(init_vals, model = "D0", sadx, sady, verbose = FALSE)
```

**Arguments**

- `init_vals`: A vector of parameter values in which to start the Maximum Likelihood algorithm, depending on the provided model:
  - model: "D0" parameters = c(theta, alpha)
  - model: "D1" parameters = c(theta, alpha X, alpha Y)
- `model`: The chosen model to calculate the maximum likelihood for, please note that the vector of parameters should contain the corresponding parameters in the right order. The user can pick one of these models:
  - "D0"
  - "D1"
- `sadx`: The Species Abundance Distribution of guild X
- `sady`: The Species Abundance Distribution of guild Y
- `verbose`: TRUE/FALSE flag, indicates whether intermediate output is shown on screen

**Value**

The output is a list containing the following:

- `par`: A vector containing the parameter values at the maximum likelihood value
- `value`: The likelihood at the corresponding parameter values
- `counts`: Number of function evaluations required
convergence
-2: invalid input
-1: number of maximum function evaluations exceeded
 0: success: convergence
 1: limit of machine precision reached

message
A character string giving a diagnostic message from the optimizer,

hessian
Hessian matrix (not implemented for this package)

Author(s)
Thijs Janzen

Examples

## Not run:
J <- 10000

theta <- 100
alpha_x <- 0.1

simul_data <- generate.Guilds(theta, alpha_x, alpha_x, J)

# initial parameters for the D0 model c(theta, alpha)
LL <- maxLikelihood.Guilds(init_vals = c(theta, alpha_x),
  model = "D0",
  sadx = simul_data$guildX,
  sady = simul_data$guildY)

## End(Not run)
maxLikelihood.Guilds.Conditional

Arguments

init_vals  init_vals corresponds to a vector of parameter values in which to start the Maximum Likelihood algorithm, depending on the provided model:
- model: "D0" parameters = c(theta, alpha)
- model: "D1" parameters = c(theta, alpha X, alpha Y)

model  The chosen model to calculate the maximum likelihood for, please note that the vector of parameters should contain the corresponding parameters in the right order. The user can pick one of these models:
- "D0"
- "D1"

sadx  The Species Abundance Distribution of guild X
sady  The Species Abundance Distribution of guild Y
verbose  TRUE/FALSE flag, indicates whether intermediate output is shown on screen

Value

The output is a list containing the following:

par  a vector containing the parameter values at the maximum likelihood value
value  the likelihood at the corresponding parameter values
counts  Number of function evaluations required
convergence  -2: invalid input
-1: number of maximum function evaluations exceeded
0: success: convergence
1: limit of machine precision reached
message  A character string giving a diagnostic message from the optimizer,
hessian  Hessian matrix (not implemented for this package)

Author(s)

Thijs Janzen

Examples

theta = 20
alpha = 0.1
initParams <- c(theta, alpha)
maxLikelihood.Guilds.Conditioned(initParams,
model = "D0",
sadx = 1:20,
sady = 1:20,
verbose = TRUE)
preston_plot

Barplot in Preston style of an abundance dataset

Description

This function first sorts abundances into octaves, and then plots the resulting distribution.

Usage

preston_plot(abund, expected, ...)

Arguments

abund       vector containing the number of individuals per species
expected    vector containing the expected number of species per octave
...         further graphical arguments that can be passed to barplot()

Author(s)

Thijs Janzen

Examples

theta = 10
m = 0.1
J = 1000
I = m * (J - 1) / (1 - m)

abund <- generate.ESF(theta, I, J)
par(mfrow = c(1,2))
preston_plot(abund)
abund.expect <- expected.SAD(theta, m, J)
preston_plot(abund, abund.expect)
Index

* **Etienne Sampling Formula**
  GUILDS-package, 2

* **GUILDS**
  GUILDS-package, 2

* **Guilds**
  generate.Guilds.Cond, 8

* **Neutral Theory**
  generate.Guilds.Cond, 8
  GUILDS-package, 2

expected.SAD, 3
expected.SAD.Guilds, 4
expected.SAD.Guilds.Conditional, 5

generate.ESF, 6
generate.Guilds, 7
generate.Guilds.Cond, 8
GUILDS (GUILDS-package), 2
GUILDS-package, 2

logLikelihood.ESF, 9
logLikelihood.Guilds, 10

maxLikelihood.ESF, 12
maxLikelihood.Guilds, 13
maxLikelihood.Guilds.Conditional, 14

preston_plot, 16