Package 'swapClass'

June 22, 2017

Type Package

Title A Null Model Adapted to Abundance Class Data in Ecology

Version 1.0.1

Date 2017-06-22

Author Benjamin Borgy, Remi Perronne, Xavier Reboud

Maintainer Benjamin Borgy <benjamin.borgy@gmail.com>

Depends R (>= 3.1.1), inline, utils, methods

Description

A null model randomizing semi-quantitative multi-classes (or ordinal) data by swapping sub-matrices while both the row and the column marginal sums are held constant.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-22 09:24:09 UTC

R topics documented:

`swapClass-package` *SwapClass : a simple null model adapted to abundance classes data*

Description

A null model randomizing semi-quantitative multi-classes (or ordinal) data by swapping sub-matrices while both the row and the column marginal sums are held constant.

Details

Package:	swapClass
Type:	Package
Version:	1.0.1
Date:	2017-06-22
License:	GPL-3

Author(s)

Benjamin Borgy (benjamin.borgy@gmail.com), Remi Perronne, Xavier Reboud

References

Borgy B. - Dynamic and assembly of weed communities: Approach by statistical modeling (2011). Ph.D. Thesis. INRA Agroicologie & Universiti de Bourgogne. DOI: 10.13140/RG.2.1.1738.1601

Examples

```
MAT=matrix(sample(0:3,50*20,replace=TRUE,prob=c(.7,.1,.1,.1)),ncol=20)

#Calculation of the nbPerm index
nbPerm(MAT)

#Generation of null communities
NULL_MATS=nullModel(MAT)

#First null community
NULL_MATS$sim[[1]]

#Number of times that each cell has been swapped for the first null community
NULL_MATS$perms[[1]]

#nbPerm index over the observed community and the 100 null communities
plot(c(nbPerm(MAT),unlist(lapply(NULL_MATS$sim,nbPerm))),type='l',ylab="nbPerm index")

#the number of each classes per row are equal
#between observed community and the first null community
f_table = function(x) table(factor(x,levels=0:3))
all(apply(MAT,1,f_table)==apply(NULL_MATS$sim[[1]],1,f_table))

#the number of each classes per column are equal
#between observed community and the first null community
all(apply(MAT,2,f_table)==apply(NULL_MATS$sim[[1]],2,f_table))
```

nbPerm *Number of swaps that can be performed*

Description

The number of permutations is calculated by summing the number of "swappable" sub-matrices per pair of abundance classes. The number of abundance classes needs to be relatively low, the swapClass algorithm being efficient on semi-quantitative multi-classes (or ordinal) variables (e.g. Braun-Blanquet- type abundance/dominance scores) but not on continuous quantitative variables.

Usage

```
nbPerm(mat)
```

Arguments

mat	an object of class matrix with abundance classes.

Details

nbPerm is the sum of "swappable" sub-matrices divided by the number of cells in the matrix (number of rows x number of columns). nbPerm calls a dynamically defined functions (swapClass_NBP) with in-lined C code compiled by setCMethod on package load.

Value

a single numeric value.

Author(s)

Benjamin Borgy (benjamin.borgy@gmail.com), Remi Perronne, Xavier Reboud

References

Borgy B. - Dynamic and assembly of weed communities: Approach by statistical modeling (2011). Ph.D. Thesis. INRA Agroicologie & Universiti de Bourgogne. DOI: 10.13140/RG.2.1.1738.1601

See Also

nullModel

Examples

```
MAT=matrix(sample(0:3,50*20,replace=TRUE,prob=c(.7,.1,.1,.1)),ncol=20)
nbPerm(MAT)
```

`nullModel` *SwapClass null model*

Description

a simple null model adapted to semi-quantitative multi-classes (or partially ordered) data.

Usage

```
nullModel(mat, nsim = 100, recursive = TRUE, burnin = NULL, thin = NULL, strata = NULL)
```

Arguments

`mat`	an object of class matrix (or data.frame) with numerical abundance classes (site by row and species by column).
`nsim`	Number of simulated null communities (positive integer), default = 100.
`recursive`	A logical value indicating if generation of a new null community is performed from the last one created, defaut = T.
`burnin`	Number of null communities discarded before proper analysis (positive integer), if NULL (defaut) then burnin = nrow(mat)*ncol(mat)*10.
`thin`	Number of discarded null communities between two evaluations (positive integer) if recursive=T, if NULL (defaut) then thin = max(nrow(mat)*ncol(mat)*10,1000).
`strata`	a numeric vector of length egal to nrow(mat) supplying strata. Swap are performed within strata, defaut = NULL.

Details

The SwapClass model is derived from the "swap philosophy" apply on presence-absence data. The limited number of abundance classes and their repetitiveness allow the extension of the "swap" method classically applied on presence-absence data with two classes (0 and 1). In the same way that "swap" methods permute sub-matrices of presence/absence community matrix, semi-quantitative data can be randomized by swapping sub-matrices while row and column marginals are not modified. nullModel calls a dynamically defined functions (swapClass_swapC) with inlined C code compiled by setCMethod on package load.

Value

`sim`	an object of class list containing the null communities
`perms`	an object of class list containing matrices with the number of times each cell has been swapped.

Author(s)

Benjamin Borgy (benjamin.borgy@gmail.com), Remi Perronne, Xavier Reboud

References

Borgy B. - Dynamic and assembly of weed communities: Approach by statistical modeling (2011). Ph.D. Thesis. INRA Agroicologie & Universiti de Bourgogne. DOI: 10.13140/RG.2.1.1738.1601

See Also

nbPerm

Examples

```
MAT=matrix(sample(0:3,50*20,replace=TRUE,prob=c(.7,.1,.1,.1)),ncol=20)

#Generation of null communities
NULL_MATS=nullModel(MAT)

#First null community
NULL_MATS$sim[[1]]

#Number of times that each cell has been swapped for the first null community
NULL_MATS$perms[[1]]

#nbPerm index over the observed community and the 100 null communities
plot(c(nbPerm(MAT),unlist(lapply(NULL_MATS$sim,nbPerm))),type='l',ylab="nbPerm index")

#the number of each classes per row are equal
#between observed community and the first null community
f_table = function(x) table(factor(x,levels=0:3))
all(apply(MAT,1,f_table)==apply(NULL_MATS$sim[[1]],1,f_table))

#the number of each classes per column are equal
#between observed community and the first null community
all(apply(MAT,2,f_table)==apply(NULL_MATS$sim[[1]],2,f_table))
```

Index