Package ‘fgpt’

February 19, 2015

Type Package

Title Floating Grid Permutation Technique

Version 2.3

Date 2015-02-19

Author Reinder Radersma & Ben Sheldon

Maintainer Reinder Radersma <reinder.radersma@zoo.ox.ac.uk>

Description A permutation technique to explore and control for spatial autocorrelation. This package contains low level functions for performing permutations and calculating statistics as well as higher level functions. Higher level functions are an easy to use function for performing spatially restricted permutation tests and summarize and plot results.

VignetteBuilder knitr

Suggests knitr

License GPL

Repository CRAN

Repository/R-Forge/Project fgm

Repository/R-Forge/Revision 48

Repository/R-Forge/DateTimeStamp 2015-02-19 05:46:51

Date/Publication 2015-02-19 07:44:06

Depends R (>= 2.10)

NeedsCompilation no

R topics documented:

fgpt-package *Floating Grid Permutation Technique*

Description

FGPT is a spatially restricted permutation technique. The package contains both low level functions to produce permuted datasets as well as higher level functions to perform spatially restricted permutation tests and functions to summarize and plot the results.

Details

`fgeasy` is the single step version of the FGPT, which is most suitable for first time users. Please read the vignette for a more extensive discription of the method and functions.

Author(s)

Reinder Radersma & Ben Sheldon

References

Reinder Radersma & Ben C. Sheldon, 2015. A new permutation technique to explore and control for spatial autocorrelation. Submitted

fgeasy *Single step version of the Floating Grid Permutation Technique*

Description

The Floating Grid Permutation Technique is a spatially restricted permutation technique. Please read the reference mentioned below and/or the vignette before using this function. `fgeasy` can be used to perform single step permutation tests. This function can be used for detecting spatial autocorrelation for univariate observations (using Moran's I), pairwise observations (using a correlation coefficient) or multivariate observations for both single and pairwise sets of multivariate observations (using mean distance or relatedness).

Usage

```
fgeasy(xy, group=1, marks, iter=999, ratio=1, scale.seq=seq(from=0, to=max(dist(xy)),
  length.out=21)[2:21], bootstrap=FALSE, pairwise=FALSE, correlate=FALSE)
```

Arguments

`xy` Geographical locations of observations.

`group` Group membership for the observations. `group` is optional.

`marks` Should be a vector containing all observations, a two-column matrix containing two paired observations or a squared matrix containing relationships or distances between all observations (such as genetic relatedness or distance). The latter can be used to perform multivariate analyses. Note that the squared matrix does not need to be symmetrical: for instance, the genetic relatedness between paired individuals can be investigated by using males as columns and females as rows. `NA`s are allowed.

`iter` Number of iterations for every grid cell size. Default is 999, though it is adviseable to perform 9999 interations for the final results.

`ratio` The ratio between the sides of the grid cells. Default is 1.

`scale.seq` Sequence of all grid cell sizes that will be tested. The number, order and values of this sequence are completely free, as long as they are numeric. Default is 20 equaly spaced out grid cell sizes from slightly larger than 0 to the maximum distance between any two individuals.

`bootstrap` `TRUE` if observations should to be drawn with replacement, `FALSE` if not. Default is `FALSE`.

`pairwise` `TRUE` if observations consist of two paired marks per observation, `FALSE` if not. Default is `FALSE`. If `pairwise==FALSE` Moran's I is used as test statistic.

`correlate` If one wants to compare two paired observations correlations between the observations can be used. The argument `correlate` can be used to pass on the peferred method of correlation as available in `cor.test`; so `"pearson"`, `"kendall"` or `"spearman"`. If `marks` is not a two-column matrix `correlate` should be `FALSE`. Default is `FALSE`.

Value

`fgeasy` returns an object of `class` "fg", which can be accessed with the functions `summary` and `plot`.

Author(s)

Reinder Radersma

References

Reinder Radersma & Ben C. Sheldon, 2015. A new permutation technique to explore and control for spatial autocorrelation. Submitted

See Also

`cor.test`

Examples

```
## A simple univariate example
##
## Produce 49 geographical locations in a regular grid.
loc.x <- rep(1:7,7)
loc.y <- rep(1:7, each=7)

## Produce 49 observations with negative spatial autocorrelation.
marks <- c(rep(c(1,2), 24),1)+rnorm(49,0,0.1)

fg1 <- fgeasy(xy=cbind(loc.x,loc.y), marks=marks, iter=99)
summary(fg1)
plot(fg1)

## An example for two paired marks
##
## Produce 20 geographical locations.
loc.x <- 1:20
loc.y <- runif(20,0,5)

## Produce 2 x 20 phenotypes.
type1 <- 11:30+runif(20,0,5)
type2 <- 11:30+runif(20,0,5)

fg2 <- fgeasy(xy=cbind(loc.x,loc.y), marks=cbind(type1,type2), iter=99,
        pairwise=TRUE, correlate="pearson")
summary(fg2)
plot(fg2)
```

---

fgperm *Floating Grid Permutation Technique*

Description

The fgperm function produces sets of permuted observations or indices using the Floating Grid Permutation Technique, which is a method for spatially restricted permutations. fgploc offers additional functionality to manipulate observations within grid cells, for instance observations could be scaled with grid cells.

Usage

```
fgperm(xy,z=1:dim(xy)[1], scale, group=1, iter=999, ratio=1, FUN=fyshuffle, ...,
  add.obs=FALSE, as.matrix=FALSE)
fgploc(xy, scale, group=1, iter=999, ratio=1, FUN=fyshuffle, FUN.mani, ...,
  marks, pass.rand=FALSE)
```

Arguments

`xy` Two-column matrix with the geographical locations of observations.

`z` Vector with the observations. If left empty `z` will be a vector of indices from 1 to the total number of geographical locations. Those indices can be later be used for calculating statistics with `fgstat`.

`scale` Value indicating the spatial scale of the permutations. `scale` should be positive.

`group` Optional group membership of observations. `group` can be used when observations are for instance collected over multiple years.

`iter` Number of iterations for every grid cell size. Default is 999. Note that in order to produce a probability an observation is assigned to any of the geographical locations is a negative function of the distance between its original and assigned location many iterations are needed.

`ratio` The ratio between the sides of the grid cells. Default is 1.

`FUN` Function to perform randomizations. Note that the function must be able to randomize one value, which is for instance a issue if using `sample`. For solutions see the Details and Examples of `sample`. The default is the Fisher-Yates shuffle `fyshuffle`.

`FUN.mani` Function to perform manipulations of the observations within grid cells. This functionality should be used together with `cal.stat` to calculate specific statistics.

`...` Optional arguments to `FUN` and `FUN.mani`.

`marks` Should either be left empty, be a vector or a matrix. When `z` contains observations `marks` should be left empty. If the randomization procedure is for testing one variable, `marks` should be a vector for which the row numbers correspond to the index values in `z`. If the randomization procedure is for testing two variables (for instance the distance between them), `marks` should be a matrix for which the row numbers correspond to the index values in `z`.

`add.obs` If `TRUE` the first set in the output will be the observed values.

`as.matrix` If `TRUE` the output will be a matrix. If `FALSE` the output will be a list, which is needed when analyzing the data with `cal.stat`.

`pass.rand` If `TRUE` The sets of randomized observations are also passed on to the output. Default is `FALSE`.

Details

Before using those functions please read the reference or vignette. Alternatively use the more user-friendly function `fgeasy`. If there are missing values for the observations, leave `z` empty and enter the observations as `marks` in the `fgstat` function.

Value

`fgperm` returns a `list` or a `matrix`, depending on the setting of argument `as.matrix`.

Author(s)

Reinder Radersma

References

Reinder Radersma & Ben C. Sheldon, 2015. A new permutation technique to explore and control for spatial autocorrelation. Submitted

Examples

```
## 200 random geographical locations
xy <- array(runif(400,0,2), dim=c(200,2))

## run fgperm to produce 99 randomizations for scale 1
test <- fgperm(xy, scale=1, iter=99, add.obs=TRUE)

## run fgperm to produce 99 bootstraps for scale 1
test <- fgperm(xy, scale=1, iter=99, FUN=function(x){
          x[sample.int(length(x),replace=TRUE)]}, add.obs=TRUE)

## 200 times 200 random distances (e.g. genetic relatedness between mated pairs)
trait <- array(rnorm(200*200,0.6,0.1), dim=c(200,200))
## make the observed pairs more alike
diag(trait) <- diag(trait)+0.02

## make two rows and two colums empty
trait[,3] <- NA
trait[,50] <- NA
trait[6,] <- NA
trait[12,] <- NA

## calculate means; will give NAs because there are missing values
calc <- fgstat(test,trait,mean)

## calculate means
calc <- fgstat(test,trait,mean, na.rm=TRUE)

## plot means
hist(calc)
abline(v=calc[1], col="red", lwd=2)
```

---

fgstat *Calculate statistics for FGPT*

Description

Calculates a set of values for a particular statistic or sets of observations, typically for observed values and multiple sets of randomized observations.

Usage

```
fgstat(rand,marks,FUN=mean, ...)
```

Arguments

`rand` A list for which the elements are either sets of randomized variables or randomized index values. `rand` can, but not necessarily is, the output of the function `fgperm`.

`marks` Should either be left empty, be a vector or a matrix. When `rand` contains randomized variables `marks` should be left empty. If the randomization procedure is for testing one variable, `marks` should be a vector for which the row numbers correspond to the index values used in `rand`. If the randomization procedure is for testing two variables (for instance the distance between them), `marks` should be a matrix for which the row numbers correspond to the index values used in `rand`.

`FUN` Any function used to calculate the statistic of interest (e.g. `mean`, `median`, `var`, `sd`). Default for `FUN` is `mean`.

`...` Optional arguments to `FUN`. A particular useful one if dealing with missing values and using one of the functions from `base` is `na.rm=TRUE`.

Details

`fgstat` is designed to calculate statistics for spatial explicit data for which randomized data sets are generated with `fgperm`.

Value

`fgstat` returns a vector of statistics. If `rand` is the output of `fgperm` and `add.obs=TRUE`, the first value is the statistic for the observed data and the rest for randomizations.

Author(s)

Reinder Radersma

References

Reinder Radersma & Ben C. Sheldon, 2015. A new permutation technique to explore and control for spatial autocorrelation. Submitted

Examples

```
#### Example for fgrand

## 200 random geographical locations
xy <- array(runif(400,0,2), dim=c(200,2))

## run fgperm to produce 99 randomizations for scale 1
test <- fgperm(xy, scale=1, iter=99, add.obs=TRUE)

## run fgperm to produce 99 bootstraps for scale 1
test <- fgperm(xy, scale=1, iter=99, FUN=function(x){
          x[sample.int(length(x),replace=TRUE)]}, add.obs=TRUE)
```

```
## 200 times 200 random distances (e.g. genetic relatedness between mated pairs)
trait <- array(rnorm(200*200,0.6,0.1), dim=c(200,200))
## make the observed pairs more alike
diag(trait) <- diag(trait)+0.02

## make two rows and two colums empty
trait[,3] <- NA
trait[,50] <- NA
trait[6,] <- NA
trait[12,] <- NA

## calculate means; will give NAs because there are missing values
calc <- fgstat(test,trait,mean)

## calculate means
calc <- fgstat(test,trait,mean, na.rm=TRUE)

## plot means
hist(calc)
abline(v=calc[1], col="red", lwd=2)
```

fyshuffle *Fisher-Yates shuffle*

Description

Function to shuffle vectors according to the Fisher-Yates procedure

Usage

```
fyshuffle(x)
```

Arguments

x	Vector containing the sequence which needs to be shuffled. This vector can be of any type and it is allowed to have one element.

Details

Other than the `sample` function `fyshuffle` treats a single value as a vector with one element and will therefore return this element as the shuffled version of the original vector (which are similar).

Value

A randomized version of the input vector.

Author(s)

Reinder Radersma

References

Durstenfeld (1964) Communications of the ACM 7(7):420

See Also

`sample`

Examples

```
x <- 1:10
fyshuffle(x)

y <- c("a","b","c","d","e","f")
fyshuffle(y)
```

`Gpulex` *Size assortative pairing in a freshwater shrimp (*Gammarus pulex*)*

Description

This is a simulated data set of geographical locations and body size measurements for two populations of fresh water Crustaceans. Both populations share the geographical locations and female phenotypes, but not the male phenotypes (`m.pheno` and `m.pheno2` respectively). The first population does not show size assortative pairing, while the second does.

Usage

```
Gpulex
```

Format

An R data object containing one two-column matrix with geographical coordinates (`Gp.xy`) and three vectors with female (`f.pheno`) and male body size measurements (`m.pheno` and `m.pheno2`).

Source

Code to produce this data set can be found in electronic supplementary materials of the following reference.

References

Reinder Radersma & Ben C. Sheldon, 2015. A new permutation technique to explore and control for spatial autocorrelation. Submitted

Pmajor *Inbreeding avoidance in the Great Tit (*Parus major*)*

Description

This is a simulated data set of geographical locations and relatedness measures for two populations of Great Tits, a small passerine. Both populations share the geographical locations, but not the relatedness tables. The first population does not show inbreeding avoidance, while the second does.

Usage

```
Pmajor
```

Format

An R data object containing one two-column matrix with geographical coordinates (`xy`) and two relatedness tables between all females and males of the populations (`rel1` and `rel2` respectively).

Source

Code to produce this data set can be found in electronic supplementary materials of the following reference.

References

Reinder Radersma & Ben C. Sheldon, 2015. A new permutation technique to explore and control for spatial autocorrelation. Submitted

Index