Package ‘overlapptest’

April 22, 2023

Type Package

Title Test Overlapping of Polygons Against Random Rotation

Version 1.3

Date 2023-04-22

Author Marcelino de la Cruz Rot

Depends R (>= 2.10), spatstat.geom

Suggests sf

Maintainer Marcelino de la Cruz <marcelino.delacruz@urjc.es>

Description Tests the observed overlapping polygon area in a collection of polygons against a null model of random rotation, as explained in De la Cruz et al. (2017) <doi:10.13140/RG.2.2.12825.72801>.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2023-04-22 17:50:02 UTC

R topics documented:

Astragalus *Astragalus and Sesleria Plants*

Description

Oulines of the individuals of *Astragalus sempervirens* and *Sesleria albicans* present in a 2 x 2 m plot in the Pyrenean montains.

Usage

```
data("Astragalus")
data("Sesleria")
```

Format

Each object is a multiple-polygon window with the format owin of **spatstat**.

Examples

```
data(Astragalus)
Astragalus
plot(Astragalus)
# total area covered by Astragalus
area.owin(Astragalus)
# number of individual polygons
length(Astragalus$bdry)
# area of each individual
sapply(Astragalus$bdry, function(x) area.owin(owin(poly=x)))
```

centroidiam *Compute Centroids and Diameters*

Description

Computes the centroid and diameter of each individual polygon in a multi-polygon owin object.

Usage

```
centroidiam(ventana1)
```

Arguments

ventana1	A multiple-polygon window with the format owin of **spatstat**

Details

Iteratively applies the functions centroid.owin and diameter of **spatstat** to each polygon in the multipolygon owin and computes its centroid and its diameter.

Value

diams	Vector of diameters
centroids	Matrix with the coordinates of the centroids

Author(s)

Marcelino de la Cruz Rot

Examples

```
data(Astragalus)
X<-centroidiam(Astragalus)
X$centroids
X$diams
```

---

check.ventana *Checks for Anticlockwise Vertices*

Description

Checks that the vertices of polygons in a multi-polygon owin object are listed anticlockwise and, if some are not, tries to correct them.

Usage

```
check.ventana(ventana)
```

Arguments

ventana	A multiple-polygon window with the format owin of **spatstat**.

Details

This functions should be employed after importing shapefiles into owin objetcs (see vignette). Vertices of the individual polygons in the multiple-polygon owin objects should be listed anticlockwise to avoid errors in the computations of area overlapp (clockwise listed polygons represent "holes" in **spatstat**). This functions checks this and, in case that the vertices of some polygons are listed clockwise, tries to revert their order.

Value

A multiple-polygon window with the format owin of **spatstat** with the vertices of all polygons listed anticockwise. The order number of the corrected polygons are included in the attribute "malos1". If there has been any polygon whose vertices have not been corrected, their orden number are included in the attribute "malos2": these polygons should be corrected manually.

Author(s)

Marcelino de la Cruz Rot

Examples

```
data(Astragalus)
# For illustrative purposes, make the vertices of some individual polygon to be listed clockwise
Astragalus.malo<-Astragalus
Astragalus.malo$bdry[[14]]<-lapply(Astragalus.malo$bdry[[14]], rev)
# check and correct
Astragalus.corrected<-(check.ventana(Astragalus.malo))
attributes(Astragalus.corrected)
```

---

pval *P Value for a Monte Carlo Test of Polygon Overlapping*

Description

Computes the p-value and sign of deviation for the hypothesis that the first value in a vector is larger or smaller (i.e., a two-sided test) than the expected value represented by all the other values in the vector, or alternatively, computes a one-sided test.

Usage

```
pval(x, alternative=c("two.sided", "less", "greater"))
```

Arguments

`x`	A vector, usually with some observed statistic in the first position (the observed overlapp) followed by a sequence of the same statistic computed for several realizations of a null model against which we test our hypothesis (i.e., a sequence of simulated, i.e., rotated, overlapps).
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less". You can specify just the initial letter.

Value

Vector of length one whose absolute value represents the p-value and whose sign indicates wether the first value in `x` is larger (positive) or smaller (negative) than the expected value.

Author(s)

Marcelino de la Cruz

Examples

```
pval(c(0,1:99))
pval(c(100,1:99))
pval(c(100,1:199))
pval(c(200,1:199))
pval(c(0,1:199))
```

---

rotawin *Rotate Individual Polygons*

Description

Randomly rotates individual polygons around their centroids.

Usage

```
rotawin(ventana)
```

Arguments

ventana	A multiple-polygon window with the format owin of **spatstat**.

Details

rotawin applies an independent random rotation to each of the polygons in a multiple-polygon owin object.

Value

rotawin returns the original owin object with the individual polygons randomly rotated.

Author(s)

Marcelino de la Cruz Rot

Examples

```
data(Astragalus)
plot(Astragalus)
plot(rotawin(Astragalus), add=TRUE, border=2)
```

`test.intersection` *Test Overlapping of Polygons Against Random Rotation*

Description

These functions test the overlaping surface area of a colection of polygons against a a null model of random rotation. `test.auto.intersection` test for overlapping between polygons of the same type (i.e., the same species) whereas `test.intersection` test for overlapping between polygons of different types (i.e., between different species). `test.auto.intersection.p` and `test.intersection.p` functions are parallelized versions which might significantly reduce computing times.

Usage

```
test.auto.intersection(ventana1, nsim = 199, win = NULL,
    prop=1, centroides1= NULL,    diametros1 =NULL)
test.intersection(ventana1, ventana2, nsim, prop = 1, win = NULL,
   centroides1 = NULL, diametros1 = NULL, centroides2 = NULL,
   diametros2 = NULL)
test.auto.intersection.p(ventana1, nsim = 199, win = NULL,
    prop=1, centroides1= NULL,    diametros1 =NULL, ncl=2)
test.intersection.p(ventana1, ventana2, nsim, prop = 1, win = NULL,
   centroides1 = NULL, diametros1 = NULL, centroides2 = NULL,
   diametros2 = NULL, ncl=2)
```

Arguments

`ventana1`	A multiple-polygon window with the format `owin` of **spatstat**. The "accessory" species in the case of `test.intersection`.
`ventana2`	A multiple-polygon window with the format `owin` of **spatstat**. The species whose polygons will be rotated (i.e., the "focal" species) in the case of `test.intersection`.
`nsim`	Number of simulations for the Monte Carlo test.
`prop`	Proportion of the diameter of each polygon which will be employed to select potential close neighbours. See details.
`win`	Observation window, employed to control for edge effects. An objetct with the format `owin` of **spatstat**. If not provided, it will be estimated by the x and y ranges from `ventana`.
`centroides1`	Matrix or `data.frame` with the coordinates of the centroids of the polygons of `ventana1`.
`diametros1`	Vector with the diameters of the polygons of `ventana1`.
`centroides2`	Matrix or `data.frame` with the coordinates of the centroids of the polygons of `ventana2`.
`diametros2`	Vector with the diameters of the polygons of `ventana2`.
`ncl`	Number of clusters for the parallel implementation.

Details

The summary statistic employed in the test is the overall sum of the overlapping areas of intersecting polygons. To reduce the computing burden, area intersections are only computed for polygon pairs for which the distance between their centoids is smaller than the sum of their largest diameters (weighted by the argument `prop`). To avoid edge effects, the test only considers polygons whose centroids are separated from the border of the observation window by a distance larger than their largest diameter. In the pairwise tests, the polygons of `ventana1` are kept fixed in their original positions (i.e., the accessory species) and the polygons of `ventana2` (i.e., the focal species) are rotated.

Value

Both `test.auto.intersection` and `test.intersection` return a vector of length `nsim+1`, with the sum of observed overlaping areas in the first position and subsequently with the sum of overlapping areas in each the simulated (i.e., randomly rotated) realizations of the null model.

Author(s)

Marcelino de la Cruz Rot

Examples

```
data(Astragalus)
data(Sesleria)

# Test overlapping between Astragalus individuals
  # Ideally nsim should be at least 199
   Astragalus.test<- test.auto.intersection(Astragalus, nsim=19)
   # Observed overlapping area
   Astragalus.test[1]
   # p-value (negative value indicates that the observed overlapping is smaller
   # than expected)
   pval(Astragalus.test)


# Test overlapping between Astragalus  and Sesleria individuals.
# Here, Sesleria is the accesory species (its individuals are kept fixed during the
# test) and Astragalus  the focal one (its individuals are rotated)
# Ideally nsim should be at least 199
   Sesleria.Astragalus.test<- test.intersection(ventana1= Sesleria,
                                                ventana2= Astragalus, nsim=19)
   # Observed overlapping area
   Sesleria.Astragalus.test[1]
   # p-value (negative value indicates that the observed overlapping is smaller
   # than expected)
   pval(Sesleria.Astragalus.test)



# Reducing computing burden when making repetitive testing
```

```
# First, put all the polygonal regions in a list, i.e.
 owins<- list(Astragalus, Sesleria)

 # compute diameters and centroids of the individual polygons
 # in each polygonal region

centroids<- list()
diams<- list()
  for ( i in 1: length(owins)){
    cd<- centroidiam(owins[[i]])
    centroids[[i]] <- cd$centroids
    diams[[i]] <- cd$diams
}

# set the number of simulations for each test
# Ideally nsim should be at least 199
online <- interactive()
Nsim <- if(online) 19 else 3

# create an array to store the results
result <- array(NA, dim=c(length(owins),length(owins),Nsim+1))

t0<-Sys.time()
for ( i in 1: length(owins)){
   for ( j in 1: length(owins)){
     cat(i,j,"\n")
       if(j!=i) result[i,j,] <- test.intersection(owins[[i]], owins[[j]], nsim=Nsim,
centroides1=centroids[[i]], diametros1=diams[[i]],
centroides2=centroids[[j]], diametros2=diams[[j]]) else
result[i,j,] <-  test.auto.intersection(owins[[i]], nsim=Nsim,
centroides1=centroids[[i]], diametros1=diams[[i]])
   }
}
 Sys.time()-t0

 # observed values (focal species in columns)
 (observed<-  t(result[,,1]))
 # p-values
 tabla.p<- apply(result,c(1,2),pval)
 (p_values <- t(tabla.p))

 # Compare with parallelized versions:

# create an array to store the result.ps
result.p<- array(NA, dim=c(length(owins),length(owins),Nsim+1))

t0<-Sys.time()
for ( i in 1: length(owins)){
   for ( j in 1: length(owins)){
     cat(i,j,"\n")
       if(j!=i) result.p[i,j,] <- test.intersection.p(owins[[i]], owins[[j]], nsim=Nsim,
```

```
centroides1=centroids[[i]], diametros1=diams[[i]],
centroides2=centroids[[j]], diametros2=diams[[j]]) else
result.p[i,j,] <-  test.auto.intersection.p(owins[[i]], nsim=Nsim,
centroides1=centroids[[i]], diametros1=diams[[i]])
   }
}
 Sys.time()-t0

 # observed values (focal species in columns)
 (observed.p<-  t(result.p[,,1]))
 # p-values
 tabla.p.p<- apply(result.p,c(1,2),pval)
 (p_values.p <- t(tabla.p.p))
```

Index