Package: spqdep (via r-universe)

October 24, 2024

Type Package

Title Testing for Spatial Independence of Qualitative Data in Cross Section

Version 0.1.3.3

Date 2024-10-10

Maintainer Fernando Lopez <fernando.lopez@upct.es>

Description Testing for Spatial Dependence of Qualitative Data in Cross Section. The list of functions includes join-count tests, Q test, spatial scan test, similarity test and spatial runs test. The methodology of these models can be found in <doi:10.1007/s10109-009-0100-1> and <doi:10.1080/13658816.2011.586327>.

LazyData true

Encoding UTF-8

License MIT + file LICENSE

Depends R (>= 3.5), methods (>= 3.5), stats (>= 3.5)

Imports broom (>= 0.7.2), dplyr (>= 1.0.2), ggplot2 (>= 3.5.1), gt (>= 0.11.1), rgeoda (>= 0.0.10-4), gtools (>= 3.8.2), gridExtra (>= 2.3), igraph (>= 1.2.5), magrittr (>= 1.5), Matrix (>= 1.2-18), purrr (>= 0.3.4), rsample (>= 0.0.8), sp (>= 1.4-5), spatialreg (>= 1.1-8), spdep (>= 1.1-3), sf (>= 0.9-3), tidyr (>= 1.1.2)

Suggests knitr, rmarkdown, bookdown

VignetteBuilder knitr

RoxygenNote 7.3.2

Language en-US

URL `https://f8l5h9.github.io/spqdep/`

NeedsCompilation no

Author Fernando Lopez [aut, cre] (<`https://orcid.org/0000-0002-5397-9748`>), Roman Minguez [aut] (<`https://orcid.org/0000-0002-0490-3181`>), Antonio Paez [aut] (<`https://orcid.org/0000-0001-6912-9919`>), Manuel Ruiz [aut] (<`https://orcid.org/0000-0001-9228-6410`>)

Date/Publication 2024-10-21 12:00:28 UTC

Additional_repositories https://cranhaven.r-universe.dev

Repository https://cranhaven.r-universe.dev

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/spqdep

RemoteSha 8831e9e6ea67d7fadc1e868059e48be80628b054

Contents

`spqdep-package` *Testing for Spatial Dependence of Qualitative Data in Cross Section.*

Description

spqdep offers the user a collection of functions to testing for spatial independence of categorical cross-section datasets.

Details

Some functionalities that have been included in **spqdep** package are:

1. Test Q

Testing for spatial dependence with the Q test based on the symbolic entropy. See references

2. Test QMap

Two statistics for the spatial analysis of qualitative data are obtained that are based on the symbolic entropy of the maps. See references

3. Spatial runs test

Testing for spatial dependence with the global test based on runs. See references

4. Local spatial runs test

Obtain the local test based on runs. See references

5. Scan test

Testing for spatial dependence with the Scan test. See references

6. Join-count tests

Testing for spatial dependence with the classical Join-count test. See references

7. Similarity test

Testing for spatial dependence with the Similarity test. See references

Datasets

(example provinces_spain, FastFood.sf) **spqdep** includes two different datasets: provinces_spain and FastFood.sf. These sets are used to illustrate the capabilities of different functions. Briefly, their main characteristics are the following

- The *FastFood.sf* A sf object with points of the localization of Fastfood restaurants in Toronto.

- The *provinces_spain* An object of the class sf with the geometry of spanish provinces and several data sets.

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz M, López FA and A Páez (2011). Comparison of Thematic Maps Using Symbolic Entropy. *International Journal of Geographical Information Science*, 26, 413-439.
- Ruiz, M., López, F., and Páez, A. (2010). Testing for spatial association of qualitative data using symbolic dynamics. *Journal of Geographical Systems*, 12(3), 281-309.0.
- Kulldorff M, Nagarwalla N. (1995). Spatial disease clusters: Detection and Inference. *Statistics in Medicine*. 14:799-810
- Jung I, Kulldorff M, Richard OJ (2010). A spatial scan statistic for multinomial data. *Statistics in Medicine*. 29(18), 1910-1918
- Páez, A, López-Hernández, FA, Ortega-García, JA, & Ruiz, M. (2016). Clustering and co-occurrence of cancer types: A comparison of techniques with an application to pediatric cancer in Murcia, Spain. *Spatial Analysis in Health Geography*, 69-90.

See Also

Useful links:

- `https://f8l5h9.github.io/spqdep/`

`Boots.sf` *Boots.sf.*

Description

A simple features object square regular lattice 16x16. from Fig. 3.3 in Upton and Fingleton (1985). In this figure, the cells coded black/white correspond to quadrats where the perennial shrub Atriplex hymenelytra is present/absent in a sample area in Death Valley, California.

Usage

```
data(Boots.sf)
```

Format

A simple features dataframe with 256 rows and 1 column:

BW A factor with two levels: Black and White

Source

Boots, B. (2003) Developing local measures of spatial association for categorical data, Journal of Geographical Systems, 5(2), 139-160. doi:10.1007/s1010900301103

References

- Boots, B. (2003). Developing local measures of spatial association for categorical data. Journal of Geographical Systems, 5(2), 139-160. doi:10.1007/s1010900301103
- Upton G., Fingleton B. (1985) Spatial data analysis by example. Volume 1: Point pattern and quantitative data. John Wiley & Sons, Chichester

Examples

```
data(Boots.sf)
summary(Boots.sf)
plot(Boots.sf)
```

`cr_symb` *A function to create symbols*

Description

This function obtains the set of symbols to get the Q-statistic

Usage

```
cr_symb(k = k, m = m)
```

Arguments

`k`	number of categories
`m`	length of the m-surrounding

Details

...

Value

A list with two types of symbols. Permutation and Combinations-totals

`p_symb`	Matrix with symbols (permutations)
`c_symb`	Matrix with symbols (combinations)

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz M, López FA, A Páez. (2010). *Testing for spatial association of qualitative data using symbolic dynamics*. Journal of Geographical Systems. 12 (3) 281-309

See Also

`m.surround`

Examples

```
# Example 1: Obtain symbols for k=2 classes and m-surroundings of size 5
symb25 <- cr_symb(2,5)
symb25$p_symb # Permutations symbols
symb25$c_symb # Combinations-totals symbols
```

`dgp.spq` *Generation of qualitative process with spatial structure*

Description

The purpose of the function `dgp.spq` is to generate a random dataset with the dimensions and spatial structure decided by the user. This function may be useful in pure simulation experiments or with the aim of showing specific properties and characteristics of a spatial qualitative dataset.

Usage

```
dgp.spq(listw = listw, p = p,  rho = rho, control = list())
```

Arguments

`listw`	A `listw` object of the class nb, knn, listw o matrix created for example by `nb2listw` from **spatialreg** package; if `nb2listw` not given, set to the same spatial weights as the `listw` argument. It can also be a spatial weighting matrix of order *(NxN)* instead of a `listw` object. Default = `NULL`.
`p`	a vector with the percentage of elements of each categories. The lengths must be the number of categories. The sum of the elements of vector must be 1.
`rho`	the level of spatial dependence (values between -1 y 1)
`control`	List of additional control arguments. See control argument section.

Details

In order to obtain categorical random variables with controlled degrees of spatial dependence, we have designed a two- stage data generating process:

Firstly, we simulate autocorrelated data using the following model:

$$Y = (I - \rho W)^{-1}\epsilon$$

where $\epsilon = N(0, 1)$ I is the $N \times N$ identity matrix, ρ is a parameter of spatial dependence, and *W* is a connectivity matrix that determines the set of spatial relationships among points.

In the second step of the data generation process, the continuous spatially autocorrelated variable Y is used to define a discrete spatial process as follows. Let b_{ij} be defined by:

$$p(Y \leq b_{ij}) = \frac{i}{j} \quad with \quad i < j$$

Let $A = \{a_1, a_2, ..., a_k\}$ and define the discrete spatial process as:

$$X_s = a_1 \quad if \quad Y_s \leq b_{1k}$$

$$X_s = a_i \quad if \quad b_{i-1k} < Y_s \leq b_{ik}$$

$$X_s = a_k \quad if \quad Y_s > b_{k-1k}$$

Value

a factor of length N (`listw` is a matrix of order *(NxN)*) with levels the first capital letters: "A", "B", ...

The description of the DGP is available in Páez et al. 2010 (pag 291) and in details section.

Control arguments

seedinit seed to generate the data sets

Author(s)

Fernando López	<fernando.lopez@upct.es>
Román Mínguez	<roman.minguez@uclm.es>
Antonio Páez	<paezha@gmail.com>
Manuel Ruiz	<manuel.ruiz@upct.es>

References

- Ruiz M, López FA, A Páez. (2010). *Testing for spatial association of qualitative data using symbolic dynamics*. Journal of Geographical Systems. 12 (3) 281-309

See Also

Q.test, Q.map.test, sp.runs.test, scan.test

Examples

```
#
N <- 100
cx <- runif(N)
cy <- runif(N)
coor <- cbind(cx,cy)
p <- c(1/6,3/6,2/6)
rho = 0.5
listw <- spdep::nb2listw(spdep::knn2nb(spdep::knearneigh(cbind(cx,cy), k = 4)))
xf <- dgp.spq(list = listw, p = p, rho = rho)

data(provinces_spain)
listw <- spdep::poly2nb(provinces_spain, queen = FALSE)
p <- c(1/6,3/6,2/6)
rho = 0.9
xf <- dgp.spq(p = p, listw = listw, rho = rho)
provinces_spain$xf <- xf
plot(provinces_spain["xf"])
```

FastFood.sf *Selection of fast food restaurants in Toronto*

Description

A simple feature (sf) dataframe containing the locations of a selection of fast food restaurants in the city of Toronto, Canada (data are from 2008). The data are projected using EPSG: 26917 (WGS 84/UTM Zone 17N).

Usage

```
data(FastFood.sf)
```

Format

A simple features object with 614 rows and 3 variables:

ID Unique identifier of record.

Class Factor with 3 types of fast food restaurants: [P]izza, [S]andwich, and [H]amburger

geometry Geometry of simple features.

Source

Ruiz et al. (2010) https://link.springer.com/article/10.1007/s10109-009-0100-1

References

- Ruiz M, López FA, A Páez. (2010). *Testing for spatial association of qualitative data using symbolic dynamics*. Journal of Geographical Systems. 12 (3) 281-309

---

jc.test *A function to compute joint-count test for binomial and multinomial (asymptotic and permutation distributions).*

Description

A function to compute joincount tests for spatial qualitative data. This function is a wrapper of `joincount.multi` and `joincount.test` in **spdep** package.

Usage

```
jc.test(formula = NULL,
               data = NULL,
               fx = NULL,
               listw = NULL,
               na.action,
               zero.policy = NULL,
               distr = "asymptotic",
               alternative = "greater",
               control =list())
```

Arguments

`formula`	a symbolic description of the factor(s).
`data`	an (optional) data frame or a sf object with points/multipolygons geometry containing the variable(s) to be tested.
`fx`	a factor or a matrix of factors in columns
`listw`	A `listw` object created for example by `nb2listw` from **spdep** package; if `nb2listw` not given, the spatial weights are built using the object given in `listw` argument (usually an `sf` object). Default = `NULL`.
`na.action`	A function (default `options("na.action")`), can also be `na.omit` or `na.exclude`. It may be necessary to set `zero.policy` to `TRUE` because this subsetting may create no-neighbour observations.
`zero.policy`	Similar to the corresponding parameter of `lagsarlm` function in **spatialreg** package. If `TRUE` assign zero to the lagged value of zones without neighbours. Default = `NULL`.
`distr`	character. Distribution type "asymptotic" (default) or "mc".
`alternative`	character string specifying the alternative hypothesis, must be one of "greater" (default), or "less".
`control`	list of additional arguments.

Value

An `spjctest` object. This type of object is a list of `htest` objects. The length of the list is the number of factor variables included in the `formula` or the number of columns in `fx`. Each element of the list can be a `jclist` object, for binomial factors, or a `jcmulti` object for multinomial factors. See `joincount.test` or `joincount.multi` for additional details.

Control arguments

`nsim`	number of permutations used in the Monte Carlo distribution. Default `nsim` = 999.
`seedinit`	seed to select the initial element during the simulations. Default `seedinit` = 1111.
`adjust.n`	default *TRUE*, if *FALSE* the number of observations is not adjusted for no-neighbour observations, if *TRU
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, *TRUE*, or *FALSE*,
`sampling`	default *nonfree*, may be *free*. See `joincount.test` for additional information.
`queen`	default *TRUE*. Defines the neighborhood criteria for *sf* objects.
`style`	defines the style for `listw`. Default = *B* (binary).
`knn`	chooses the number of neighboors when this criteria is used. Default `knn` = 5.

Author(s)

Fernando López `<fernando.lopez@upct.es>`
Román Mínguez `<roman.minguez@uclm.es>`
Antonio Páez `<paezha@gmail.com>`

Manuel Ruiz <manuel.ruiz@upct.es>

References

- Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion, pp. 19-20.
- Upton, G., Fingleton, B. 1985 *Spatial data analysis by example: point pattern and qualitative data*, Wiley, pp. 158–170.

See Also

print.summary.spqtest, joincount.test, joincount.multi

Examples

```
## Case 1
## Multinomial + Binomial using a sf multipolygon

data("provinces_spain")
# sf::sf_use_s2(FALSE)
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
provinces_spain$Older <- cut(provinces_spain$Older, breaks = c(-Inf,19,22.5,Inf))
levels(provinces_spain$Older) = c("low","middle","high")
f1 <- ~ Older + Mal2Fml
jc1 <- jc.test(formula = f1,
               data = provinces_spain,
               distr = "mc",
               alternative = "greater",
               zero.policy = TRUE)
summary(jc1)

provinces_spain$Coast <- factor(provinces_spain$Coast)
levels(provinces_spain$Coast) = c("no","yes")
f2 <- ~ Mal2Fml + Coast
jc2 <- jc.test(formula = f2,
               data = provinces_spain,
               distr = "mc",
               zero.policy = TRUE)
summary(jc2)


# Case 2:
## Multinomial using a sf multipoint
data("FastFood.sf")
# sf::sf_use_s2(FALSE)
f1 <- ~ Type
jc3 <- jc.test(formula = f1,
               data = FastFood.sf,
               distr = "asymptotic",
               control = list(knn = 6))
 summary(jc3)

# Examples function joincount.test
data(oldcol, package = "spdep")
HICRIME <- cut(COL.OLD$CRIME, breaks = c(0,35,80), labels = c("low","high"))
names(HICRIME) <- rownames(COL.OLD)
jc4 <- jc.test(fx = HICRIME,
               listw = spdep::nb2listw(COL.nb,
               style="B"))
summary(jc4)
 jc5 <- jc.test(fx = HICRIME,
               listw = spdep::nb2listw(COL.nb, style="B"),
               distr = "mc")
 summary(jc5)
 HICRIME <- cut(COL.OLD$CRIME, breaks = c(0, 35, 80),
               labels = c("low","high"))
 names(HICRIME) <- rownames(COL.OLD)
 jc6 <- jc.test(fx = HICRIME,
               listw = spdep::nb2listw(COL.nb,
                                 style = "B"))
summary(jc6)
```

`local.jc.test` *A function to calculate the local Join Count tests.*

Description

This function calculates the local Join Count tests.This function is a wrapper of `local_joincount` in **rgeoda** package.

Usage

```
local.jc.test(formula = NULL, data = NULL, fx = NULL, case = NULL, coor = NULL,
 listw = listw, nsim = 999, control = list())
```

Arguments

`formula`	An (optional) formula with the factor included in `data`
`data`	An (optional) data frame or a sf object containing the variable to testing for.
`fx`	An (optional) factor of observations with the same length as the neighbours list in `listw`
`case`	level of the factor used to obtain the local Join Count test. See details.
`coor`	(optional) a 2xN vector with spatial coordinates. Used when *data* is not a spatial object.
`listw`	A neighbourhood list (an object type knn, listw or nb).

`nsim`	The number of permutation to obtain the local Join Count tests. Default value 999.
`control`	Optional argument. See Control argument section.

Details

This test develop by Anselin and Li (2019) can be apply only for binary variables (usually named B and W), and count the number of joins of type B-B where the proportion of observations with B is lower than half. The interest lies in identifying co-occurrences of uncommon events, i.e., situations where observations that take on the value of B constitute much less than half of the sample.
The statistic is defined as:

$$BB_i = x_i \sum_j w_{ij} x_j$$

only meaningful for those observations where $x_i = 1$, since for $x_i = 0$ the result will always equal zero.

The object `listw` can be the class:

- `knn`: Objects of the class knn that consider the neighbours in order of proximity.
- `nb`: If the neighbours are obtained from an sf object, the code internally will call the function `nb2nb_order` it will order them in order of proximity of the centroids.
- `matrix`: If a object of matrix class based in the inverse of the distance in introduced as argument, the function `nb2nb_order` will also be called internally to transform the object the class matrix to a matrix of the class nb with ordered neighbours.

Value

The output is an object of the class localjc

`local.SRQ` A matrix with

`nn`	number of neighbourhood in the localization 'i'.
`ljc`	local Join Count statistics.
`pseudo.value`	p-value of local jc tests.

Control arguments

`seedinit`	Numerical value for the seed in boot version. Default value seedinit = 123

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Anselin, L., and Li, X. (2019). Operational local join Count statistics for cluster detection. *Journal of Geographical Systems, 21(2), 189-210.*

See Also

`local.sp.runs.test`, `dgp.spq`

Examples

```
# Case 1: Local spatial runs test based on knn
N <- 100
cx <- runif(N)
cy <- runif(N)
coor <- cbind(cx,cy)
listw <- spdep::knearneigh(coor, k = 4)
p <- c(1/2,1/2)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)
ljc <- local.jc.test(fx = fx, coor = coor, case= "A", listw = listw)
print(ljc)
plot(ljc, coor = coor, sig = 0.1)


# Case 2:Fastfood example. sf (points)
data("FastFood.sf")
# sf::sf_use_s2(FALSE)
x <- sf::st_coordinates(sf::st_centroid(FastFood.sf))
listw <- spdep::knearneigh(x, k = 6)
formula <- ~ Type
ljc <- local.jc.test(formula = formula, data = FastFood.sf, case = "H",listw = listw)
print(ljc)
plot(ljc, sf = FastFood.sf, sig = 0.05)

# Case 3: With a sf object (poligons)
fname <- system.file("shape/nc.shp", package="sf")
nc <- sf::st_read(fname)
listw <- spdep::poly2nb(as(nc,"Spatial"), queen = FALSE)
p <- c(1/3,2/3)
rho = 0.5
```

```
nc$fx <- dgp.spq(p = p, listw = listw, rho = rho)
plot(nc["fx"])
formula <- ~ fx
ljc <- local.jc.test(formula = formula, data = nc, case = "A", listw = listw)
ljc
plot(ljc, sf = nc)

# Case 4: With isolated areas
data(provinces_spain)
# sf::sf_use_s2(FALSE)
listw <- spdep::poly2nb(as(provinces_spain,"Spatial"), queen = FALSE)
provinces_spain$Mal2Fml<- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
formula <- ~ Mal2Fml
ljc <- local.jc.test(formula = formula, data = provinces_spain, listw = listw)
print(ljc)
plot(ljc, sf = provinces_spain, sig = 0.1)

# Case 5: Regular lattice
data(Boots.sf)
listw <- spdep::poly2nb(as(Boots.sf,"Spatial"), queen = TRUE)
formula <- ~ BW
ljc <- local.jc.test(formula = formula, data = Boots.sf, case="B", listw = listw)
print(ljc)
plot(ljc, sf = Boots.sf, sig = 0.05)
lsr <- local.sp.runs.test(formula = formula, data = Boots.sf,
distr = "bootstrap", nsim= 99, listw = listw)
print(lsr)
plot(lsr, sf = Boots.sf, sig = 0.1)
scan <- scan.test(formula = formula, data = Boots.sf, nsim = 299,
dist = "bernoulli", nv = 60 , case = "B", windows = "elliptic")
plot(scan, sf = Boots.sf)
```

local.sp.runs.test *A function to calculate the local spatial runs tests.*

Description

This function calculates the local spatial runs tests for all localizations.

Usage

```
local.sp.runs.test(formula = NULL, data = NULL, fx = NULL,
distr = "asymptotic", listw = listw, alternative = "two.sided" , nsim = NULL,
control = list())
```

Arguments

`formula`	An (optional) formula with the factor included in `data`
`data`	An (optional) data frame or a sf object containing the variable to testing for.
`fx`	An (optional) factor of observations with the same length as the neighbors list in `listw`
`distr`	a character string specifying the distribution "asymptotic" (default) or "bootstrap"
`listw`	A neighbourhood list (an object type knn or nb) or a W matrix that indicates the order of the elements in each \$m_i-environment\$ (for example of inverse distance). To calculate the number of runs in each \$m_i-environment\$, an order must be established, for example from the nearest neighbour to the furthest one.
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
`nsim`	Default value is NULL to obtain the asymptotic version of the local test. For the bootstrap version nsim is the number of permutations to obtain the pseudo-value.
`control`	Optional argument. See Control Argument section.

Details

The object `listw` can be the class:

- `knn`: Objects of the class knn that consider the neighbours in order of proximity.
- `nb`: If the neighbours are obtained from an sf object, the code internally will call the function `nb2nb_order` it will order them in order of proximity of the centroids.
- `matrix`: If a object of matrix class based in the inverse of the distance in introduced as argument, the function `nb2nb_order` will also be called internally to transform the object the class matrix to a matrix of the class nb with ordered neighbours.

Value

The output is an object of the class localsrq

`local.SRQ` A matrix with

`runs.i`	number of runs in the localization 'i'.
`E.i`	expectation of local runs statistic in the localization 'i'.
`Sd.i`	standard deviate of local runs statistic in the localization 'i'.
`z.value`	standard value of local runs statistic (only for asymptotic version).
`p.value`	p-value of local local runs statistic (only for asymptotic version).
`zseudo.value`	standard value of local runs statistic (only for boots version).
`pseudo.value`	p-value of local runs statistic (only for boots version).

`MeanNeig` Mean of run.i
`MaxNeig` Maximum of run.i

`listw` the object `listw`
`alternative` a character string describing the alternative hypothesis

Control arguments

`seedinit`	Numerical value for the seed in boot version. Default value seedinit = 123

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

@references

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *Working paper*.

See Also

`sp.runs.test`, `dgp.spq`

Examples

```
# Case 1: Local spatial runs test based on knn
N <- 100
cx <- runif(N)
cy <- runif(N)
x <- cbind(cx,cy)
listw <- spdep::knearneigh(cbind(cx,cy), k = 10)
p <- c(1/6,3/6,2/6)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)

# Asymtotic version
lsrq <- local.sp.runs.test(fx = fx, listw = listw, alternative = "less")
print(lsrq)
plot(lsrq, sig = 0.05)
# Asymtotic version
lsrq <- local.sp.runs.test(fx = fx, listw = listw, alternative = "two.sided",
                           distr ="bootstrap", nsim = 399)
print(lsrq)
plot(lsrq, sig = 0.1)
```

```
# Case 2:Fastfood example. sf (points)
data("FastFood.sf")
# sf::sf_use_s2(FALSE)
x <- sf::st_coordinates(sf::st_centroid(FastFood.sf))
listw <- spdep::knearneigh(x, k = 10)
formula <- ~ Type
lsrq <- local.sp.runs.test(formula = formula, data = FastFood.sf, listw = listw)
print(lsrq)
plot(lsrq, sf = FastFood.sf, sig = 0.05)


# Case 3: With a sf object (poligons)
fname <- system.file("shape/nc.shp", package="sf")
nc <- sf::st_read(fname)
listw <- spdep::poly2nb(as(nc,"Spatial"), queen = FALSE)
p <- c(1/6,3/6,2/6)
rho = 0.5
nc$fx <- dgp.spq(p = p, listw = listw, rho = rho)
plot(nc["fx"])
formula <- ~ fx
lsrq <- local.sp.runs.test(formula = formula, data = nc, listw = listw)
print(lsrq)
plot(lsrq, sf = nc)
# Version boot
lsrq <- local.sp.runs.test(formula = formula, data = nc, listw = listw,
                           distr ="bootstrap", nsim = 399)
print(lsrq)
plot(lsrq, sf = nc)

# Case 4: With isolated areas
data(provinces_spain)
listw <- spdep::poly2nb(as(provinces_spain,"Spatial"), queen = FALSE)
provinces_spain$Mal2Fml<- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
plot(provinces_spain["Mal2Fml"])
formula <- ~ Mal2Fml
lsrq <- local.sp.runs.test(formula = formula, data = provinces_spain, listw = listw)
print(lsrq)
plot(lsrq, sf = provinces_spain, sig = 0.1)

# Boots Version
lsrq <- local.sp.runs.test(formula = formula, data = provinces_spain, listw = listw,
                           distr ="bootstrap", nsim = 199)
print(lsrq)
plot(lsrq, sf = provinces_spain, sig = 0.10)

# Case 5: SRQ test based on a distance matrix (inverse distance)

N <- 100
cx <- runif(N)
cy <- runif(N)
coor <- as.data.frame(cbind(cx,cy))
coor <- sf::st_as_sf(coor,coords = c("cx","cy"))
```

```
n = dim(coor)[1]
dis <- 1/matrix(as.numeric(sf::st_distance(coor,coor)), ncol = n, nrow = n)
diag(dis) <- 0
dis <- (dis < quantile(dis,.10))*dis
p <- c(1/6,3/6,2/6)
rho <- 0.5
fx <- dgp.spq(p = p, listw = dis, rho = rho)
lsrq <- local.sp.runs.test(fx = fx, listw = dis)
print(lsrq)
plot(lsrq, coor = cbind(cx,cy), sig = 0.05)
lsrq <- local.sp.runs.test(fx = fx, listw = dis, data = )
print(lsrq)
plot(lsrq, sf = coor)
# Version boots
lsrq <- local.sp.runs.test(fx = fx, listw = dis, data = coor,
                           distr ="bootstrap", nsim = 299)
print(lsrq)
plot(lsrq, sf = coor)

# SRQ test based on inverse distance
data("FastFood.sf")
# sf::sf_use_s2(FALSE)
n = dim(FastFood.sf)[1]
dis <- 1000000/matrix(as.numeric(
          sf::st_distance(FastFood.sf, FastFood.sf)),
          ncol = n, nrow = n)
diag(dis) <- 0
dis <- (dis < quantile(dis,.01))*dis
formula <- ~ Type
lsrq <- local.sp.runs.test(formula = formula, data = FastFood.sf, listw = dis)
print(lsrq)
# plot(lsrq, sf = FastFood.sf)
```

m.surround — *A function to generate m-surroundings*

Description

This function obtains the m-surroundings by selecting the *m-1* nearest neighbors of each observation, allowing for a degree of overlap of *r*.

Usage

```
m.surround(x, m, r = 1, distance = "Euclidean", control = list())
```

Arguments

x	input sf object with points/multipolygons geometry or matrix of spatial coordinates

`m`	dimension of m-surrounding
`r`	maximum overlapping between any two m-surroundings.
`distance`	character. For Cartesian coordinates only: one of Euclidean, Hausdorff or Frechet; for geodetic coordinates, Great Circle distances are computed. (see `sf::st_distance()`). Default = "Euclidean".
`control`	List of additional control arguments.

Value

A list of class `list` and `m_surr` containing the following components:

`ms`	a matrix. Each row is a m-surrounding.
`R`	total number of observations.
`rowexcl`	index of rows excluded.
`mdtms`	distances between the observations of each m-surrounding.
`N`	total number of symbolized observations.
`m`	length of the m-surroundings.
`r`	overlapping degree.
`initobs`	element to start the generation of m-surroundings.
`distance`	type of distance.
`m`	length of the m-surroundings.
`x`	the input "x" as sf-object.

Control arguments

- `initobs`: Initial observation to begin the m-surrounding process. Default = 1.
- `dtmaxabs`: Threshold of distance (in absolute value) to prune the m-surroundings. Any m-surrounding exceeding the threshold is excluded. If `dtmaxabs` = 0 there is no exclusion of m-surroundings. Default = 0.
- `dtmaxpc`: Threshold of distance (as a percentage of the maximum distance between observations) to prune the m-surroundings. Any m-surrounding exceeding the threshold is excluded. Example if `dtmaxpc` = 0.1 the m-surrounding exceeding the 10 If `dtmaxpc` = 0 there is no exclusion of m-surroundings. Default = 0.
- `dtmaxknn`: Eliminate m-surroundings where some of the elements are not among the closest knn (k-nearest-neighbors). Example, if `dtmaxknn` = 4 exclude m-surroundings where some of the elements are not between the 4 closest. Default `dtmaxknn` = 0 (no exclusion)

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz M, López FA, A Páez. (2010). Testing for spatial association of qualitative data using symbolic dynamics. *Journal of Geographical Systems*. 12 (3) 281-309

Examples

```
# Example 1: Obtain m-surroundings with degree of overlapping r
N <- 100
cx <- runif(N)
cy <- runif(N)
x <- cbind(cx,cy)
m <- 3
r <- 1
msurr_points <- m.surround(x = x, m = m, r = r)
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
summary(msurr_points)
msurr_points <- m.surround(x = x, m = m, r = r,
                control = list(dtmaxpc = 0.1))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
summary(msurr_points)
msurr_points <- m.surround(x = x, m = m, r = r,
                   control = list(dtmaxknn = 20))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
summary(msurr_points)

# Example 2:

data("FastFood.sf")
m <- 3
r <- 1
msurr_points <- m.surround(x = FastFood.sf, m = m, r = r,
                           distance = "Euclidean",
                           control = list(dtmaxpc = .001))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
print(msurr_points)
summary(msurr_points)
msurr_points <- m.surround(x = FastFood.sf, m = m, r = r,
                           distance = "Euclidean",
                           control = list(dtmaxknn = 20))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
summary(msurr_points)


# Example 3: With isolated areas
data(provinces_spain)
# sf::sf_use_s2(FALSE)
plot(sf::st_geometry(provinces_spain))
```

```
m <- 3
r <- 1
msurr_points <- m.surround(x = provinces_spain, m = m, r = r,
                           distance = "Euclidean",
                           control = list(dtmaxknn = 8))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)

# Example 4: Examples with multipolygons
fname <- system.file("shape/nc.shp", package="sf")
nc <- sf::st_read(fname)
plot(sf::st_geometry(nc))
m <- 3
r <- 1
msurr_polygonsf <- m.surround(x = nc, m = m, r = r,
                   distance = "Great Circle",
                   control=list(dtmaxpc = 0.20))
plot(msurr_polygonsf, type = 1)
plot(msurr_polygonsf, type = 2)

# Example 5: With regular lattice
sfc = sf::st_sfc(sf::st_polygon(list(rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0)))))
hexs <- sf::st_make_grid(sfc, cellsize = 0.1, square = FALSE)
hexs.sf <- sf::st_sf(hexs)
listw  <- spdep::poly2nb(as(hexs.sf, "Spatial"), queen = FALSE)
m <- 3
r <- 1
msurr_polygonsf <- m.surround(x = hexs.sf, m = m, r = r)
plot(msurr_polygonsf, type = 1)
plot(msurr_polygonsf, type = 2)
summary(msurr_polygonsf)
```

methods_localjc *Methods for class localjc*

Description

The `plot()` function allows the user to plot significant observations. The `print()` function is used to print the number of runs in each localization. Additional information of expected values and standard deviation, z-value ans p-value is prited for each observation.

Usage

```
## S3 method for class 'localjc'
print(x, ...)

## S3 method for class 'localjc'
plot(x, ..., sf = NULL, coor = NULL, sig = 0.05)
```

Arguments

x	a localjc object created by Q.test.
...	further arguments passed to or from other methods.
sf	optional argument for plot() method to include a sf object (default = NULL)
coor	optional argument for plot() method to include coordinates of points (default = NULL)
sig	significant level for each observation in plot() method. Default sig = 0.05

Value

No return value, called for side effects

Author(s)

Fernando López	<fernando.lopez@upct.es>
Román Mínguez	<roman.minguez@uclm.es>
Antonio Páez	<paezha@gmail.com>
Manuel Ruiz	<manuel.ruiz@upct.es>

References

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *working paper*.

Examples

```
# Example 1: Local spatial runs test based on knn
N <- 100
cx <- runif(N)
cy <- runif(N)
x <- cbind(cx,cy)
listw <- spdep::knearneigh(cbind(cx,cy), k = 10)
p <- c(1/6,3/6,2/6)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)

# Asymtotic version
lsrq <- local.sp.runs.test(fx = fx, listw = listw, alternative = "less")
print(lsrq)
plot(lsrq, sig = 0.05)
```

methods_localsrq *Methods for class localsrq*

Description

The plot() function allows the user to plot significant observations. The print() function is used to print the number of runs in each localization. Additional information of expected values and standard deviation, z-value ans p-value is prited for each observation.

Usage

```
## S3 method for class 'localsrq'
print(x, ...)

## S3 method for class 'localsrq'
plot(x, ..., sf = NULL, coor = NULL, sig = 0.05)
```

Arguments

x	a localsrq object created by Q.test.
...	further arguments passed to or from other methods.
sf	optional argument for plot() method to include a sf object (default = NULL)
coor	optional argument for plot() method to include coordinates of points (default = NULL)
sig	significant level for each observation in plot() method. Default sig = 0.05

Value

No return value, called for side effects

Author(s)

Fernando López	<fernando.lopez@upct.es>
Román Mínguez	<roman.minguez@uclm.es>
Antonio Páez	<paezha@gmail.com>
Manuel Ruiz	<manuel.ruiz@upct.es>

References

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *working paper*.

Examples

```
# Example 1: Local spatial runs test based on knn
N <- 100
cx <- runif(N)
cy <- runif(N)
x <- cbind(cx,cy)
listw <- spdep::knearneigh(cbind(cx,cy), k = 10)
p <- c(1/6,3/6,2/6)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)

# Asymtotic version
lsrq <- local.sp.runs.test(fx = fx, listw = listw, alternative = "less")
print(lsrq)
plot(lsrq, sig = 0.05)
```

`methods_m_surr` *Method for class m_surr*

Description

A function to plots the m-surrounds give an object of the class *m_surr* obtain with the code `m.surround`.
The `plot()` function allows the user view the configuration of the m-surroundings.
The argument `type` select the type o visualization.
The `print()` print the matrix of the m-surrounding.
. The `summary` give information about the characteristics of the m-surroundings.
.

Usage

```
## S3 method for class 'm_surr'
summary(object, ...)

## S3 method for class 'm_surr'
plot(x, ..., type = 1)

## S3 method for class 'm_surr'
print(x, ...)
```

Arguments

`object`	object of class *m_surr*. 2 plot W matrix with network
`...`	further arguments passed to or from other methods.
`x`	object of class *m_surr*
`type`	numeric. 1 (default) to get the plot with igraph.

Value

No return value, called for side effects

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *Working paper*.

Examples

```
# Example 1: Obtain m-surroundings with degree of overlapping r
N <- 100
cx <- runif(N)
cy <- runif(N)
x <- cbind(cx,cy)
m = 4
r = 2
msurr_points <- m.surround(x = x, m = m, r = r,control = list(dtmaxabs = 0.5))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
print(msurr_points)

# Example 2:
data("FastFood.sf")
m = 6
r = 1
msurr_points <-  m.surround(x = FastFood.sf, m = m, r = r, distance = "Euclidean",
                            control = list(dtmaxpc = .2))
plot(msurr_points, type = 1)
plot(msurr_points, type = 2)
print(msurr_points)
```

methods_qmap *Method for class qmap*

Description

A function to plot the difference in frequencies of symbols of each map. The `plot()` function to obtain the plot. The argument `ci` select the confidence level.

Arguments

`x`	object of class *qmap*
`ci`	confidence level for the difference of probabilities of symbols in ‘plot‘ method. Default `ci = 0.95`
`...`	further arguments passed to or from other methods.

Author(s)

Fernando López <fernando.lopez@upct.es>
Román Mínguez <roman.minguez@uclm.es>
Antonio Páez <paezha@gmail.com>
Manuel Ruiz <manuel.ruiz@upct.es>

References

- Ruiz M, López FA and A Páez (2011). Comparison of Thematic Maps Using Symbolic Entropy. *International Journal of Geographical Information Science*, 26, 413-439.
- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *working paper*.

Examples

```
# Example 1:
N <- 100
cx <- runif(N)
cy <- runif(N)
coor <- cbind(cx,cy)
p <- c(1/6,3/6,2/6)
rho = 0.5
listw <- spdep::nb2listw(spdep::knn2nb(spdep::knearneigh(cbind(cx,cy), k = 4)))
fx <- dgp.spq(list = listw, p = p, rho = rho)
q.test <- Q.test(fx = fx, coor = coor, m = 3, r = 1)
plot(q.test)
```

methods_scantest *Methods for class scantest*

Description

The plot() function allows the user plot the significant cluster(s).
summary list information about the most likelihood cluster.

Usage

```
## S3 method for class 'scantest'
plot(x, ..., sf = NULL, coor = NULL)

## S3 method for class 'scantest'
summary(object, ...)
```

Arguments

x	is a scantest object created by scan.test for plot() method.
...	further arguments passed to or from other methods.
sf	optional argument for plot() method to include a sf object (default = NULL)
coor	optional argument for plot() method to include coordinates of points (default = NULL)
object	a scantest object created by scan.test for summary() method.

Value

No return value, called for side effects

Author(s)

Fernando López	<fernando.lopez@upct.es>
Román Mínguez	<roman.minguez@uclm.es>
Antonio Páez	<paezha@gmail.com>
Manuel Ruiz	<manuel.ruiz@upct.es>

References

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *Working paper*.

`methods_spqtest` *Methods for class spqtest*

Description

The `plot()` function allows the user obtain the plot of relative frequency of each symbol (standard and equivalent) with the confidence interval. The `print()` function is used to get a list with the values of Q test for standard symbolization based on permutations and for equivalent symbolization based on combinations. `summary()` print a table with the output of the Q test.

Usage

```
## S3 method for class 'spqtest'
plot(x, ..., ci = 0.95)
```

Arguments

`x`	a `spqtest` object created by `Q.test`.
`...`	further arguments passed to or from other methods
`ci`	confidence level for the intervals in `plot` method. Default `ci = 0.95`

Value

This functions does not return any value

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *working paper*.

`nb2nb_order`	*A function to order the elements of the m_i-subrrounds*

Description

An auxiliary function. In the case of obtaining the list of neighbors of class nb or poly2nb, it is necessary to reorder the elements based on distance and/or angle.

Usage

```
nb2nb_order(listw = listw, sf = NULL)
```

Arguments

`listw`	an object of the nb class.
`sf`	the sf object used to get the `listw` .

Details

Sort the elements of a list nb. First by distance and

Value

An object of the nb class with the elements in order.

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paez@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

@references

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *Working paper*.

See Also

`dgp.spq`, `sp.runs.test`, `local.sp.runs.test`

Examples

```
# With a sf object (irregular lattice)
library(sf)
fname <- system.file("shape/nc.shp", package="sf")
nc <- sf::st_read(fname)
listw <- spdep::poly2nb(as(nc,"Spatial"), queen = FALSE)
listw.order <- nb2nb_order(listw = listw, sf = nc)

# With a sf object (regular lattice: hexagons)
sfc = sf::st_sfc(sf::st_polygon(list(rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0)))))
hexs <- sf::st_make_grid(sfc, cellsize = 0.1, square = FALSE)
hexs.sf <- sf::st_sf(hexs)
listw  <- spdep::poly2nb(as(hexs.sf, "Spatial"), queen = FALSE)
listw.order <- nb2nb_order(listw = listw, sf = hexs.sf)
```

`Newark.sf` *Extract of 1880 US Census for Newark, New Jersey.*

Description

A simple features object with geocoded information about respondents in the 1880 US Census with selected demographic information coded as dummy variables. The data are projected using EPSG: 32618 (WGS 84/UTM Zone 18N). The coordinates have been and jiggled to create unique coordinates for each observation.

Usage

```
data(Newark.sf)
```

Format

A simple features dataframe with 21,520 rows and 8 columns:

ID Unique identifier of record.

YANKEE Dummy variable for ethnicity of respondent: 1 if Yankee, 0 otherwise.

IRISH Dummy variable for ethnicity of respondent: 1 if Irish, 0 otherwise.

GERMAN Dummy variable for ethnicity of respondent: 1 if German, 0 otherwise.

under30 Dummy variable for age of respondent: 1 if younger than 30 years old, 0 otherwise.

mar Dummy variable for marital status of respondent: 1 if married, 0 otherwise.

usborn Dummy variable for place of birth of respondent: 1 if born in the US, 0 otherwise.

geometry geometry of the simple features object

Source

Páez et al. (2012) doi:10.1080/00045608.2011.620502

References

- Paez, A., Ruiz, M., Lopez, F. & Logan, J. (2012). *Measuring Ethnic Clustering and Exposure with the Q Statistic: An Exploratory Analysis of Irish, Germans, and Yankees in 1880 Newark.*. Annals of the Association of American Geographers.

Examples

```
data(Newark.sf)
summary(Newark.sf)
```

`plot.sprunstest` *Plot the empirical distribution of runs*

Description

Plot the empirical distribution of runs

Usage

```
## S3 method for class 'sprunstest'
plot(x, ...)
```

Arguments

`x`	A object of class *sprunstest*.
`...`	further arguments passed to or from other methods.

Details

Plot the histogram with the empirical distribution of the runs

Value

No return value, called for side effects

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paez@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

See Also

sp.runs.test.

Examples

```
# Example 1: Fastfood example. sf (points)
data("FastFood.sf")
x <- sf::st_coordinates(sf::st_centroid(FastFood.sf))
listw <- spdep::knearneigh(x, k = 2)
formula <- ~ Type
srq <- sp.runs.test(formula = formula, data = FastFood.sf, listw = listw, nsim = 299)
plot(srq)

# Example 2: Spain example (poligons with 0 neinghbourhood)
data("provinces_spain")
sf::sf_use_s2(FALSE)
listw <- spdep::poly2nb(as(provinces_spain,"Spatial"), queen = FALSE)
provinces_spain$Older <- cut(provinces_spain$Older, breaks = c(-Inf,19,22.5,Inf))
levels(provinces_spain$Older) = c("low","middle","high")
formula <- ~ Older
srq <- sp.runs.test(formula = formula, data = provinces_spain, listw = listw, nsim = 299)
plot(srq)
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
formula <- ~ Mal2Fml
srq <- sp.runs.test(formula = formula, data = provinces_spain, listw = listw, nsim = 299)
plot(srq)
```

print.summary.spjctest

Print method for objects of class summary.spjctest.

Description

Print method for objects of class summary.spjctest.

Usage

```
## S3 method for class 'summary.spjctest'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	object of class *summary.spjctest*.
digits	number of digits to show in printed tables. Default: max(3L, getOption("digits") - 3L).
...	further arguments passed to or from other methods.

Value

No return value, called for side effects

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

See Also

`summary.spqtest`.

`print.summary.spqtest` *Print method for objects of class summary.spqtest.*

Description

Print method for objects of class summary.spqtest.

Usage

```
## S3 method for class 'summary.spqtest'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

`x`	object of class *summary.spqtest*.
`digits`	number of digits to show in printed tables. Default: max(3L, getOption("digits") - 3L).
`...`	further arguments passed to or from other methods.

Value

No return value, called for side effects

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

See Also

`summary.spqtest`.

`provinces_spain` *Provinces in Spain.*

Description

A simple features object with the provinces in Spain and selected demographic and economic information.

Usage

```
data(provinces_spain)
```

Format

A simple features dataframe with 50 rows and 15 columns:

province Names of provinces in Spain as factor

CCAA Names of Autonomous Communities in Spain as factor

ID_INE National Institute of Statistics unique identifier of the provinces

Population Population in the province in 2020

Density Population density in the province in persons/km^2

Older Percentage of population 65 and older in the provice in 2020

Median_Age Median age of population in the province in 2020

Mal2Fml Ratio of male to female population in the province in 2020

GDPpc GDP per capita in the province in 2016

Transit Dummy variable for mass transit system in province; 1: YES

Area Area of the province

Altitude Altitude of the province

Coast A dummy variable that indicates whether the province is in the coast; 1: YES

Meteo_Station Identifier of meteorological station representative of the province used to retrieve climatic variables

geometry geometry of the simple features object

Source

Instituto Nacional de Estadistica http://www.ine.es/

Climatic data: Agencia Estatal de Meteorologia http://www.aemet.es/

Páez et al. (2020)

References

- Paez, A., Lopez, F.A., Menezes, T., Cavalcanti, R., & Pitta, M. (2020). A Spatio-Temporal Analysis of the Environmental Correlates of COVID-19 Incidence in Spain. *Geographical Analysis*. 53(3) 397-421

Examples

```
data(provinces_spain)
summary(provinces_spain)
```

`Q.map.test` *Compute the QE and QI tests of Equivalence and Independence between maps*

Description

This function compute the QE and QI tests for maps comparison based on symbolic entropy.

Usage

```
Q.map.test(formula = formula, data = data, coor = NULL, m = m, r = 1,
type = "combinations", control = list())
```

Arguments

`formula`	a symbolic description of the two factors.
`data`	(optional) a data frame or a sf object containing the variables to testing for.
`coor`	(optional) a 2xN vector with coordinates.
`m`	length of m-surrounding.
`r`	maximum overlapping between any two m-surroundings (default = 1).
`type`	Type of symbols: "permutations" or "combinations". Default "combinations"
`control`	Optional argument. See Control Argument section.

Details

If `data` is not a sf object the `coor` argument with the coordinates of each observation must be included.

Value

A list with two objects of the class `htest`. The first one is the QE test of Equivalence between maps and the second one is the QI test of independence between maps. the elements of each test are:

`method`	a character string giving description of the method.
`data.name`	a character string giving the name(s) of the data.
`statistic`	the value of the statistic QE or/and QI.
`alternative`	a character string describing the alternative hypothesis.
`p.value`	p-value for QE or QI.
`parameter`	free degree of the statistic for QE or QI.
`symb`	A matrix with the symbols.
`mh`	m-surrounding of th map.
`Tm`	number of maps (ONLY 2).
`sample.size`	number of symbolized observations.
`nsk`	a matrix Tm x symbols with the frequency of the number of symbols of each map.

Control arguments

Several parameters to construct the m-surrounding

dtmaxabs Delete degenerate surrounding based on the absolute distance between observations.

dtmaxpc A value between 0 and 1. Delete degenerate surrounding based on the distance. Delete m-surrounding when the maximum distance between observation is upper than k percentage of maximum distance between anywhere observation.

dtmaxknn A integer value 'k'. Delete degenerate surrounding based on the near neighborhood criteria. Delete m-surrounding is a element of the m-surrounding is not include in the set of k near neighborhood of the first element

seedinit seed to select the initial element to star the algorithm to compute the m-surroundings.

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz M, López FA and A Páez (2011). Comparison of Thematic Maps Using Symbolic Entropy. *International Journal of Geographical Information Science*, 26, 413-439.
- Ruiz, M., López, FA, and Páez, A. (2010). Testing for spatial association of qualitative data using symbolic dynamics. *Journal of Geographical Systems*, 12(3), 281-309.0.

See Also

dgp.spq, m.surround, Q.test

Examples

```
# Case 1:
N <- 200
cx <- runif(N)
cy <- runif(N)
x <- cbind(cx,cy)
listw <- spdep::nb2listw(spdep::knn2nb(
           spdep::knearneigh(cbind(cx,cy), k = 4)))
p <- c(1/6, 3/6, 2/6)
rho = 0.5
QY1 <- dgp.spq(p = p, listw = listw, rho = rho)
rho = 0.8
QY2 <- dgp.spq(p = p, listw = listw, rho = rho)
dt = data.frame(QY1,QY2)
m = 3
r = 1
formula <- ~ QY1 + QY2
control <- list(dtmaxknn = 10)
qmap <- Q.map.test(formula = formula, data = dt, coor = x, m = m, r = r,
                   type ="combinations", control = control)
print(qmap)
plot(qmap)
plot(qmap, ci=.6)
plot(qmap[[1]]$mh)
summary(qmap[[1]]$mh)

control <- list(dtmaxknn = 20)
qmap <- Q.map.test(formula = formula, data = dt, coor = x, m = m, r = r,
                   type ="permutations", control = control)
print(qmap)
plot(qmap)
plot(qmap[[1]]$mh)
qmap <- Q.map.test(formula = formula, data = dt, coor = x, m = m, r = r,
                   type ="combinations")
print(qmap)
plot(qmap)
control <- list(dtmaxknn = 10)
qmap <- Q.map.test(formula = formula, data = dt, coor = x, m = m, r = r,
                   type ="combinations", control = control)
print(qmap)
plot(qmap)

# Case 2:
data(provinces_spain)
# sf::sf_use_s2(FALSE)
m = 3
r = 1
```

```
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
provinces_spain$Coast <- factor(provinces_spain$Coast)
levels(provinces_spain$Coast) = c("no","yes")
formula <- ~ Coast + Mal2Fml
qmap <- Q.map.test(formula = formula, data = provinces_spain, m = m, r = r,
                   type ="combinations")
print(qmap)
plot(qmap)
plot(qmap[[1]]$mh)

control <- list(dtmaxknn = 6)
qmap <- Q.map.test(formula = formula, data = provinces_spain, m = m, r = r,
                   type ="combinations", control = control)
print(qmap)
plot(qmap[[1]]$mh)
summary(qmap[[1]]$mh)
```

`Q.test` *A function to compute Q test for spatial qualitative data*

Description

A function to compute Q test for spatial qualitative data.

Usage

```
Q.test(formula = NULL, data = NULL, na.action,
fx = NULL, coor = NULL, m = 3, r = 1, distr = "asymptotic",
control = list())
```

Arguments

`formula`	a symbolic description of the factor(s).
`data`	an (optional) data frame or a sf object with points/multipolygons geometry containing the variable(s) to be tested.
`na.action`	action with NA values
`fx`	a factor or a matrix of factors in columns
`coor`	(optional) a 2xN vector with spatial coordinates. Used when *data* is not a spatial object
`m`	length of m-surrounding (default = 3).
`r`	only for asimtotic distribution. Maximum overlapping between any two m-surroundings (default = 1).
`distr`	character. Distribution type "asymptotic" (default) or "mc".
`control`	Optional argument. See Control Argument section.

Details

The Q-test is a simple, consistent, and powerful statistic for qualitative spatial independence that we develop using concepts from symbolic dynamics and symbolic entropy. The Q test can be used to detect, given a spatial distribution of events, patterns of spatial association of qualitative variables in a wide variety of settings.
The Q(m) statistic was introduced by Ruiz et al. (2010) as a tool to explore geographical co-location/co-occurrence of qualitative data. Consider a spatial variable X which is the result of a qualitative process with a set number of categorical outcomes a_j (j=1,...,k). The spatial variable is observed at a set of fixed locations indexed by their coordinates s_i (i=1,..., N), so that at each location si where an event is observed, X_i takes one of the possible values a_j.

Since the observations are georeferenced, a spatial embedding protocol can be devised to assess the spatial property of co-location. Let us define, for an observation at a specified location, say s_0, a surrounding of size m, called an m-surrounding.
The m-surrounding is the set of m-1 nearest neighbours from the perspective of location s_0. In the case of distance ties, a secondary criterion can be invoked based on direction.
Once that an embedding protocol is adopted and the elements of the m-surrounding for location s_0 have been determined, a string can be obtained that collects the elements of the local neighborhood (the m-1 nearest neighbors) of the observation at s_0. The m-surrounding can then be represented in the following way:

$$X_m(s_0) = (X_{s_0}, X_{s_1}, ...X_{s_{m-1}})$$

Since each observation Xs takes one of k possible values, and there are m observations in the m-surrounding, there are exactly k possible unique ways in which those values can co-locate. This is the number of permutations with replacement.
For instance, if k=2 (e.g. the possible outcomes are a1=0 and a2=1) and m=3, the following eight unique patterns of co-location are possible (the number of symbols is n_σ=8): (0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1), and (1,1,1). Each unique co-locationtype can be denoted in a convenient way by means of a symbol σ_i $(i = 1, 2, ..., k^m)$. It follows that each site can be uniquely associated with a specific symbol, in a process termed symbolization. In this way, we say that a location s is of type σ_i if and only if $X_m(s) = \sigma_i$.
Equivalent symbols (see Páez, et al. 2012) can be obtained by counting the number of occurrences of each category within an m-surrounding. This surrenders some topological information (ordering within the m-surrounding is lost) in favor of a more compact set of symbols, since the number of combinations with replacement.

Definition of Q(m) statistic

Let $\{X_s\}_{s \in R}$ be a discrete spatial process and m be a fixed embedding dimension. The statistic Q testing the null hypothesis:

$H_0 : \{X_s\}_{s \in R}$ is spatially independent, against any other alternative.

For a fixed $m \geq 2$, the relative frequency of symbols can be used to define the symbolic entropy of the spatial process as the Shanon entropy of the distinct symbols:

$$h(m) = -\sum_j p_{\sigma_j} ln(p_{\sigma_j})$$

where

$$p_{\sigma_j} = \frac{n_{\sigma_j}}{R}$$

with n_{σ_j} is simply the number of times that the symbol σ_j is observed and R the number of symbolized locations. The entropy function is bounded between $0 < h(m) \leq \eta$.

The Q statistic is essentially a likelihood ratio test between the symbolic entropy of the observed pattern and the entropy of the system under the null hypothesis of a random spatial sequence:

$$Q(m) = 2R(\eta - h(m))$$

with $\eta = ln(k^m)$. The statistic is asymptotically χ^2 distributed with degrees of freedom equal to the number of symbols minus one.

Value

An list of two object of the class `htest`. Each element of the list return the:

`data.name`	a character string giving the names of the data.
`statistic`	Value of the Q test
`N`	total number of observations.
`R`	total number of symbolized observations.
`m`	length m-surrounding.
`r`	degree of overlapping.
`df`	degree of freedom.
`distr`	type of distribution used to get the significance of the Q test.
`type`	type of symbols.

Control arguments

distance character to select the type of distance. Default = "Euclidean" for Cartesian coordinates only: one of Euclidean, Hausdorff or Frechet; for geodetic coordinates, great circle distances are computed (see sf::st_distance())

dtmaxabs Delete degenerate surrounding based on the absolute distance between observations.

dtmaxpc A value between 0 and 1. Delete degenerate surrounding based on the distance. Delete m-surrounding when the maximum distance between observation is upper than k percentage of maximum distance between anywhere observation.

dtmaxknn A integer value 'k'. Delete degenerate surrounding based on the near neighbourhood criteria. Delete m-surrounding is a element of the m-surrounding is not include in the set of k near neighbourhood of the first element

nsim number of simulations for get the Monte Carlo distribution. Default = 999

seedinit seed to select the initial element to star the algorithm to get compute the m-surroundings or to start the simulation

Standard-Permutation vs Equivalent-Combination Symbols

The symbolization protocol proposed by Ruiz et al. (2010) - call these Standard-Permutation Symbols — contains a large amount of topological information regarding the units of analysis, including proximity and direction. In this sense, the protocol is fairly general. On the other hand, it is easy to see that the combinatorial possibilities can very quickly become unmanageable. For a process with k = 3 outcomes and m = 5, the number of symbols becomes $3^5 = 243$; for k = 6 and m = 4 it is $6^4 = 1,296$. Depending on the number of observations N, the explosion in the number of symbols can very rapidly consume degrees of freedom for hypothesis testing, because as a rule of thumb it is recommended that the number of symbolized locations be at least five times the number of symbols used (e.g., $R \geq 5k^m$), and R will usually be a fraction of N.

As an alternative, we propose a symbolization protocol that sacrifices some amount of topological detail for conciseness. The alternative is based on the standard scheme; however, instead of retaining proximity and direction relationships, it maintains only the total number of occurrences of each outcome in an m-surrounding. We call these Equivalent-Combination Symbols. Because order in the sequence is not considered in this protocol, instead of a permutation with repetition, the number of symbols reflects a combination with repetition.

Selection of m-surrounding with Controlled Degree of Overlapping (r)

To select S locations for the analysis, coordinates are selected such that for any two coordinates s_i , s_j the number of overlapping nearest neighbours of s_i and s_j are at most r. The set S, which is a subset of all the observations N, is defined recursively as follows. First choose a location s_0 at random and fix an integer r with $0 \leq r < m$. The integer r is the degree of overlap, the maximum number of observations that contiguous m-surroundings are allowed to have in common.
Let $\{s_1^0, s_2^0, ..., s_{m-1}^0\}$ be the set of nearest neighbours to s_0, where the s_i^0 are ordered by distance to s_0, or angle in the case of ties. Let us call $s_1 = s_{m-r-1}^0$ and define $A_0 = \{s_0, s_0^1, ..., s_{m-r-2}^0\}$. Take the set of nearest neighbours to s_1, namely, $\{s_1^1, s_2^1, ..., s_{m-1}^1\}$ in the set of locations $S \setminus A_0$ and define $s_2 = s_{m-r-1}^1$. Nor for i>1 we define $s_i = s_{m-r-1}^{i-1}$ where s_{m-r-1}^{i-1} is in the ser of nearest neighbors to s_{i-1}, $\{s_1^{i-1}, s_2^{i-1}, ..., s_{m-1}^{i-1}\}$,of the set $S \setminus \{\cup_{j=0}^{i-1} A_j\}$. Continue this process while there are locations to symbolize.

Selection of m-surroundings for bootstrap distribution

The bootstrapped-based testing can provide an advantage since overlapping between m-surroundings is not a consideration, and the full sample can be used.

Author(s)

Fernando López <fernando.lopez@upct.es>
Román Mínguez <roman.minguez@uclm.es>
Antonio Páez <paezha@gmail.com>
Manuel Ruiz <manuel.ruiz@upct.es>

References

- Ruiz M, López FA, A Páez. (2010). Testing for spatial association of qualitative data using symbolic dynamics. *Journal of Geographical Systems*. 12 (3) 281-309
- López, FA, and A Páez. (2012). Distribution-free inference for Q(m) based on permutational bootstrapping: an application to the spatial co-location pattern of firms in Madrid *Estadística Española*, 177, 135-156.

Examples

```
# Case 1: With coordinates
N <- 200
cx <- runif(N)
cy <- runif(N)
coor <- cbind(cx,cy)
p <- c(1/6,3/6,2/6)
rho = 0.3
listw <- spdep::nb2listw(spdep::knn2nb(spdep::knearneigh(cbind(cx,cy), k = 4)))
fx <- dgp.spq(list = listw, p = p, rho = rho)
q.test <- Q.test(fx = fx, coor = coor, m = 3, r = 1)
summary(q.test)
plot(q.test)
print(q.test)

q.test.mc <- Q.test(fx = fx, coor = coor, m = 3, distr = "mc", control = list(nsim = 999))
summary(q.test.mc)
plot(q.test.mc)
print(q.test.mc)


# Case 2: With a sf object
data("FastFood.sf")
f1 <- ~ Type
q.test <- Q.test(formula = f1, data = FastFood.sf, m = c(3, 4),
r = c(1, 2, 3), control = list(distance ="Euclidean"))
summary(q.test)
plot(q.test)
print(q.test)

# Case 3: With a sf object with isolated areas
data("provinces_spain")
sf::sf_use_s2(FALSE)
provinces_spain$Mal2Fml<- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
provinces_spain$Older <- cut(provinces_spain$Older, breaks = c(-Inf,19,22.5,Inf))
levels(provinces_spain$Older) = c("low","middle","high")
f1 <- ~ Older + Mal2Fml
q.test <- Q.test(formula = f1,
data = provinces_spain, m = 3, r = 1, control = list(seedinit = 1111))
summary(q.test)
```

```
print(q.test)
plot(q.test)
q.test.mc <- Q.test(formula = f1, data = provinces_spain, m = 4, r = 3, distr = "mc",
control = list(seedinit = 1111))
summary(q.test.mc)
print(q.test.mc)
plot(q.test.mc)

# Case 4: Examples with multipolygons
library(sf)
fname <- system.file("shape/nc.shp", package="sf")
nc <- sf::st_read(fname)
qb79 <- quantile(nc$BIR79)
nc$QBIR79 <- (nc$BIR79 > qb79[2]) + (nc$BIR79 > qb79[3]) +
(nc$BIR79 >= qb79[4]) + 1
nc$QBIR79 <- as.factor(nc$QBIR79)
plot(nc["QBIR79"], pal = c("#FFFEDE","#FFDFA2", "#FFA93F", "#D5610D"),
     main = "BIR79 (Quartiles)")
sid79 <- quantile(nc$SID79)
nc$QSID79 <- (nc$SID79 > sid79[2]) + (nc$SID79 > sid79[3]) +
(nc$SID79 >= sid79[4]) + 1
nc$QSID79 <- as.factor(nc$QSID79)
plot(nc["QSID79"], pal = c("#FFFEDE","#FFDFA2", "#FFA93F", "#D5610D"),
     main = "SID79 (Quartiles)")
f1 <- ~ QSID79 + QBIR79
lq1nc <- Q.test(formula = f1, data = nc, m = 5, r = 2,
control = list(seedinit = 1111, dtmaxpc = 0.5, distance = "Euclidean") )
print(lq1nc)

lq2nc <- Q.test(formula = f1, data = nc, m = 5, r = 2,
control = list(dtmaxpc = 0.2) )
print(lq2nc)

lq3nc <- Q.test(formula = f1, data = nc, m = 5, r = 2,
control = list(dtmaxknn = 5) )
print(lq3nc)

# Case 5: Examples with points and matrix of variables
fx <- matrix(c(nc$QBIR79, nc$QSID79), ncol = 2, byrow = TRUE)
mctr <- suppressWarnings(sf::st_centroid(st_geometry(nc)))
mcoor <- st_coordinates(mctr)[,c("X","Y")]
q.test <- Q.test(fx = fx, coor = mcoor, m = 5, r = 2,
                 control = list(seedinit = 1111, dtmaxpc = 0.5))
print(q.test)
plot(q.test)
```

scan.test   *Compute the scan test*

Description

This function compute the spatial scan test for Bernoulli and Multinomial categorical spatial process, and detect spatial clusters

Usage

```
scan.test(formula = NULL, data = NULL, fx = NULL, coor = NULL, case = NULL,
nv = NULL, nsim = NULL, distr = NULL, windows = "circular", listw = NULL,
alternative = "High", minsize = 1, control = list())
```

Arguments

`formula`	a symbolic description of the factor (optional).
`data`	an (optional) data frame or a sf object containing the variable to testing for.
`fx`	a factor (optional).
`coor`	(optional) coordinates of observations.
`case`	Only for Bernoulli distribution. A element of factor, there are cases and non-cases for testing for cases versus non-cases
`nv`	Maximum windows size, default nv = N/2. The algorithm scan for clusters of geographic size between 1 and the upper limit (nv) defined by the user.
`nsim`	Number of permutations.
`distr`	distribution of the spatial process: "bernoulli" for two levels or "multinomial" for three or more levels.
`windows`	a string to select the type of cluster "circular" (default) of "elliptic".
`listw`	only for flexible windows. A neighbours list (an object of the class listw, nb or knn frop spdep) or an adjacency matrix.
`alternative`	Only for Bernoulli spatial process. A character string specifying the type of cluster, must be one of "High" (default), "Both" or "Low".
`minsize`	Minimum number of observations inside of Most Likely Cluster and secondary clusters.
`control`	List of additional control arguments.

Details

Two alternative sets of arguments can be included in this function to compute the scan test:

- `Option 1`: A factor (fx) and coordinates (coor).
- `Option 2`: A sf object (data) and the formula to specify the factor. The function consider the coordinates of the centroids of the elements of th sf object.

The spatial scan statistics are widely used in epidemiology, criminology or ecology. Their purpose is to analyse the spatial distribution of points or geographical regions by testing the hypothesis of spatial randomness distribution on the basis of different distributions (e.g. Bernoulli, Poisson or Normal distributions). The `scan.test` function obtain the scan statistic for two relevant distributions related with categorical variables: the Bernoulli and Multinomial distribution.

The spatial scan statistic is based on the likelihood ratio test statistic and is formulated as follows:

$$\Delta = \frac{\max_{z \in Z, H_A} L(\theta|z)}{\max_{z \in Z, H_0} L(\theta|z)}$$

where Z represents the collection of scanning windows constructed on the study region, H_A is an alternative hypothesis, H_0 is a null hypothesis, and $L(\theta|z)$ is the likelihood function with parameter θ given window Z
. The null hypothesis says that there is no spatial clustering on the study region, and the alternative hypothesis is that there is a certain area with high (or low) rates of outcome variables. The null and alternative hypotheses and the likelihood function may be expressed in different ways depending on the probability model under consideration.
To test independence in a spatial process, under the null, the type of windows is irrelevant but under the alternative the elliptic windows can to identify with more precision the cluster.

For big data sets (N ») the windows = "elliptic" can be so slowly

Bernoulli version

When we have dichotomous outcome variables, such as cases and noncases of certain diseases, the Bernoulli model is used. The null hypothesis is written as

$$H_0 : p = q \;\; for \;\; all \;\; Z$$

and the alternative hypothesis is

$$H_A : p \neq q \;\; for \;\; some \;\; Z$$

where p and q are the outcome probabilities (e.g., the probability of being a case) inside and outside scanning window Z, respectively. Given window Z, the test statistic is:

where cz and nz are the numbers of cases and observations (cases and noncases) within z, respectively, and C and N are the total numbers of cases and observations in the whole study region, respectively.

$$\Delta =$$

Multinomial version of the scan test

The multinomial version of the spatial scan statistic is useful to investigate clustering when a discrete spatial variable can take one and only one of k possible outcomes that lack intrinsic order information. If the region defined by the moving window is denoted by Z, the null hypothesis for the statistic can be stated as follows:

$$H_0 : p_1 = q_1; p_2 = q_2; ...; p_k = q_k$$

where p_j is the probability of being of event type j inside the window Z, and q_j is the probability of being of event type j outside the window. The alternative hypothesis is that for at least one type event the probability of being of that type is different inside and outside of the window.

The statistic is built as a likelihood ratio, and takes the following form after transformation using the natural logarithm:

$$\Delta = \max_{Z}\{\sum_{j}\{S_j^Z log(\frac{S_j^Z}{S^Z}) + (S_j - S_j^Z)log(\frac{S_j - S_j^Z}{S - S^Z})\}\} - \sum_{j} S_j log(\frac{S_j}{S})$$

where S is the total number of events in the study area and S_j is the total number of events of type j. The superscript Z denotes the same but for the sub-region defined by the moving window.
The theoretical distribution of the statistic under the null hypothesis is not known, and therefore significance is evaluated numerically by simulating neutral landscapes (obtained using a random spatial process) and contrasting the empirically calculated statistic against the frequency of values obtained from the neutral landscapes. The results of the likelihood ratio serve to identify the most likely cluster, which is followed by secondary clusters by the expedient of sorting them according to the magnitude of the test. As usual, significance is assigned by the analyst, and the cutoff value for significance reflects the confidence of the analyst, or tolerance for error.
When implementing the statistic, the analyst must decide the shape of the window and the maximum number of cases that any given window can cover. Currently, analysis can be done using circular or elliptical windows.
Elliptical windows are more time consuming to evaluate but provide greater flexibility to contrast the distribution of events inside and outside the window, and are our selected shape in the analyses to follow. Furthermore, it is recommended that the maximum number of cases entering any given window does not exceed 50% of all available cases.

Value

A object of the *htest* and *scantest* class

`method`	The type of test applied ().
`fx`	Factor included as input to get the scan test.
`MLC`	Observations included into the Most Likelihood Cluster (MLC).
`statistic`	Value of the scan test (maximum Log-likelihood ratio).
`N`	Total number of observations.
`nn`	Windows used to get the cluster.
`nv`	Maximum number of observations into the cluster.
`data.name`	A character string giving the name of the factor.
`coor`	coordinates.
`alternative`	Only for bernoulli spatial process. A character string describing the alternative hypothesis select by t
`p.value`	p-value of the scan test.
`cases.expect`	Expected cases into the MLC.
`cases.observ`	Observed cases into the MLC.
`nsim`	Number of permutations.
`scan.mc`	a (nsim x 1) vector with the loglik values under bootstrap permutation.
`secondary.clusters`	a list with the observations included into the secondary clusters.
`loglik.second`	a vector with the value of the secondary scan tests (maximum Log-likelihood ratio).
`p.value.secondary`	a vector with the p-value of the secondary scan tests.
`Alternative.MLC`	A vector with the observations included in another cluster with the same loglik than MLC.

Control arguments

`seedinit`	Numerical value for the seed (only for boot version). Default value seedinit=123

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Kulldorff M, Nagarwalla N. (1995). Spatial disease clusters: Detection and Inference. *Statistics in Medicine*. 14:799-810
- Jung I, Kulldorff M, Richard OJ (2010). A spatial scan statistic for multinomial data. *Statistics in Medicine*. 29(18), 1910-1918
- Páez, A., López-Hernández, F.A., Ortega-García, J.A., Ruiz, M. (2016). Clustering and co-occurrence of cancer types: A comparison of techniques with an application to pediatric cancer in Murcia, Spain. *Spatial Analysis in Health Geography*, 69-90.
- Tango T., Takahashi K. (2005). A flexibly shaped spatial scan statistic for detecting clusters, *International Journal of Health Geographics* 4:11.

See Also

`local.sp.runs.test`, `dgp.spq`, `Q.test`,

Examples

```
# Case 1: Scan test bernoulli
data(provinces_spain)
sf::sf_use_s2(FALSE)
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
formula <- ~ Mal2Fml
scan <- scan.test(formula = formula, data = provinces_spain, case="men",
nsim = 99, distr = "bernoulli")
print(scan)
summary(scan)
plot(scan, sf = provinces_spain)

## With maximum number of neighborhood
scan <- scan.test(formula = formula, data = provinces_spain, case = "woman",
nsim = 99, distr = "bernoulli")
```

```
print(scan)
plot(scan, sf = provinces_spain)


## With elliptic windows
scan <- scan.test(formula = formula, data = provinces_spain, case = "men", nv = 25,
nsim = 99, distr = "bernoulli", windows ="elliptic")
print(scan)
scan <- scan.test(formula = formula, data = provinces_spain, case = "men", nv = 15,
nsim = 99, distr = "bernoulli", windows ="elliptic", alternative = "Low")
print(scan)
plot(scan, sf = provinces_spain)

# Case 2: scan test multinomial
data(provinces_spain)
provinces_spain$Older <- cut(provinces_spain$Older, breaks = c(-Inf,19,22.5,Inf))
levels(provinces_spain$Older) = c("low","middle","high")
formula <- ~ Older
scan <- scan.test(formula = formula, data = provinces_spain, nsim = 99, distr = "multinomial")
print(scan)
plot(scan, sf = provinces_spain)

# Case 3: scan test multinomial
data(FastFood.sf)
sf::sf_use_s2(FALSE)
formula <- ~ Type
scan <- scan.test(formula = formula, data = FastFood.sf, nsim = 99,
distr = "multinomial", windows="elliptic", nv = 50)
print(scan)
summary(scan)
plot(scan, sf = FastFood.sf)

# Case 4: DGP two categories
N <- 150
cx <- runif(N)
cy <- runif(N)
listw <- spdep::knearneigh(cbind(cx,cy), k = 10)
p <- c(1/2,1/2)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)
scan <- scan.test(fx = fx, nsim = 99, case = "A", nv = 50, coor = cbind(cx,cy),
distr = "bernoulli",windows="circular")
print(scan)
plot(scan)

# Case 5: DGP three categories
N <- 200
cx <- runif(N)
cy <- runif(N)
listw <- spdep::knearneigh(cbind(cx,cy), k = 10)
p <- c(1/3,1/3,1/3)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)
```

```
scan <- scan.test(fx = fx, nsim = 19, coor = cbind(cx,cy), nv = 30,
distr = "multinomial", windows = "elliptic")
print(scan)
plot(scan)

# Case 6: Flexible windows
data(provinces_spain)
sf::sf_use_s2(FALSE)
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
formula <- ~ Mal2Fml
listw <- spdep::poly2nb(provinces_spain, queen = FALSE)
scan <- scan.test(formula = formula, data = provinces_spain, case="men", listw = listw, nv = 6,
                  nsim = 99, distr = "bernoulli", windows = "flexible")
print(scan)
summary(scan)
plot(scan, sf = provinces_spain)
```

similarity.test *Compute the similarity test.*

Description

This function compute the nonparametric test for spatial independence using symbolic analysis for categorical/qualitative spatial process.

Usage

```
similarity.test(formula = NULL, data = NULL, fx = NULL, listw = listw,
alternative = "two.sided", distr = "asymptotic", nsim = NULL, control = list())
```

Arguments

formula	a symbolic description of the factor (optional).
data	an (optional) data frame or a sf object containing the variable to testing for.
fx	a factor (optional).
listw	a listw object
alternative	a character string specifying the type of cluster, must be one of "High" (default), "Both" or "Low".
distr	A string. Distribution of the test "asymptotic" (default) or "bootstrap"
nsim	Number of permutations.
control	List of additional control arguments.

Details

This testing approach for spatial independence that extends some of the properties of the join count statistic. The premise of the tests is similar to the join count statistic in that they use the concept of similarity between neighbouring spatial entities (Dacey 1968; Cliff and Ord 1973, 1981). However, it differs by taking into consideration the number of joins belonging locally to each spatial unit, rather than the total number of joins in the entire spatial system. The approach proposed here is applicable to spatial and network contiguity structures, and the number of neighbors belonging to observations need not be constant.

We define an equivalence relation $\sim$ in the set of locations S. An equivalence relation satisfies the following properties:

Reflexive: $s_i \sim s_i$ for all $s_i \in S$,
Symmetric: If $s_i \sim s_j$, then $s_j \sim s_i$ for all $s_i,\ s_j \in S$ and
Transitive: If $s_i \sim s_j$ and $s_j \sim s_k$, then $s_i \sim s_k$ for all $s_i,\ s_j,\ s_k \in S$

We call the relation $\sim$ a similarity relation. Then, the null hypothesis that we are interested in is

$$H_0 : \{X_s\}_{s\in S}\ \ is\ \ iid$$

Assume that, under the null hypothesis, $P(s_i \sim s_{ji}) = p_i$ for all $s_{ji} \in N_{s_i}$.
Define

$$I_{ij} = 1\ \ if\ \ s_i \sim s_{ji}\ \ ;0\ \ otherwise$$

Then, for a fixed degree d and for all location si with degree d, the variable d

$$\Lambda_{(d,i)} = \sum_{j=1}^{d} I_{ij}$$

gives the number of nearest neighbours to si that are similar to si. Therefore, under the null hypothesis, H_0, $\Lambda(d,i)$ follows a binomial distribution $B(d, p_i)$.

Value

A object of the *htest*

`data.name`	a character string giving the names of the data.
`statistic`	Value of the similarity test
`N`	total number of observations.
`Zmlc`	Elements in the Most Likelihood Cluster.
`alternative`	a character string describing the alternative hypothesis.
`p.value`	p-value of the similarity test
`similiarity.mc`	values of the similarity test in each permutation.

Control arguments

`seedinit`	Numerical value for the seed (only for boot version). Default value seedinit=123

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Farber, S., Marin, M. R., & Paez, A. (2015). Testing for spatial independence using similarity relations. *Geographical Analysis*. 47(2), 97-120.

See Also

`sp.runs.test`, `dgp.spq`, `Q.test`, , `scan.test`

Examples

```
# Case 1:
N <- 100
cx <- runif(N)
cy <- runif(N)
listw <- spdep::knearneigh(cbind(cx,cy), k = 3)
p <- c(1/4,1/4,1/4,1/4)
rho <- 0.5
fx <- dgp.spq(p = p, listw = listw, rho = rho)
W <- (spdep::nb2mat(spdep::knn2nb(listw)) >0)*1
similarity <- similarity.test(fx = fx, data = FastFood.sf, listw = listw)
print(similarity)

# Case 2: test with formula, a sf object (points) and knn
data("FastFood.sf")
coor <- sf::st_coordinates(sf::st_centroid(FastFood.sf))
listw <- spdep::knearneigh(coor, k = 4)
formula <- ~ Type
similarity <- similarity.test(formula = formula, data = FastFood.sf, listw = listw)
print(similarity)

# Case 3:
data(provinces_spain)
```

```
listw <- spdep::poly2nb(as(provinces_spain,"Spatial"), queen = FALSE)
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
formula <- ~ Mal2Fml
similarity <- similarity.test(formula = formula, data = provinces_spain, listw = listw)
print(similarity)
```

`sp.runs.test` *Compute the global spatial runs test.*

Description

This function compute the global spatial runs test for spatial independence of a categorical spatial data set.

Usage

```
sp.runs.test(formula = NULL, data = NULL, fx = NULL,
listw = listw, alternative = "two.sided" ,
distr = "asymptotic", nsim = NULL,control = list())
```

Arguments

`formula`	a symbolic description of the factor (optional).
`data`	an (optional) data frame or a sf object containing the variable to testing for.
`fx`	a factor (optional).
`listw`	A neighbourhood list (type knn or nb) or a W matrix that indicates the order of the elements in each $m_i - environment$ (for example of inverse distance). To calculate the number of runs in each $m_i - environment$, an order must be established, for example from the nearest neighbour to the furthest one.
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
`distr`	A string. Distribution of the test "asymptotic" (default) or "bootstrap".
`nsim`	Number of permutations to obtain pseudo-value and confidence intervals (CI). Default value is NULL to don‘t get CI of number of runs.
`control`	List of additional control arguments.

Details

The order of the neighbourhoods ($m_i - environments$) is critical to obtain the test.
To obtain the number of runs observed in each $m_i - environment$, each element must be associated with a set of neighbours ordered by proximity. Three kinds of lists can be included to identify $m_i - environments$:

- `knn`: Objects of the class knn that consider the neighbours in order of proximity.

- `nb`: If the neighbours are obtained from an sf object, the code internally will call the function `nb2nb_order` it will order them in order of proximity of the centroids.
- `matrix`: If a object of matrix class based in the inverse of the distance in introduced as argument, the function `nb2nb_order` will also be called internally to transform the object the class matrix to a matrix of the class nb with ordered neighbours.

Two alternative sets of arguments can be included in this function to compute the spatial runs test:

`Option 1`	A factor (fx) and a list of neighborhood (`listw`) of the class knn.
`Option 2`	A sf object (data) and formula to specify the factor. A list of neighbourhood (listw)

Value

A object of the *htest* and *sprunstest* class

`data.name`	a character string giving the names of the data.
`method`	the type of test applied ().
`SR`	total number of runs
`dnr`	empirical distribution of the number of runs
`statistic`	Value of the homogeneity runs statistic. Negative sign indicates global homogeneity
`alternative`	a character string describing the alternative hypothesis.
`p.value`	p-value of the SRQ
`pseudo.value`	the pseudo p-value of the SRQ test if nsim is not NULL
`MeanNeig`	Mean of the Maximum number of neighborhood
`MaxNeig`	Maximum number of neighborhood
`listw`	The list of neighborhood
`nsim`	number of boots (only for boots version)
`SRGP`	nsim simulated values of statistic.
`SRLP`	matrix with the number of runs for eacl localization.

Definition of spatial run

In this section define the concepts of spatial encoding and runs, and construct the main statistics necessary for testing spatial homogeneity of categorical variables. In order to develop a general theoretical setting, let us consider $\{X_s\}_{s \in S}$ to be the categorical spatial process of interest with Q different categories, where S is a set of coordinates.

Spatial encoding: For a location $s \in S$ denote by $N_s = \{s_1, s_2 ..., s_{n_s}\}$ the set of neighbours according to the interaction scheme W, which are ordered from lesser to higher Euclidean distance with respect to location s.
The sequence as $X_{s_i}, X_{s_i+1}, ..., , X_{s_i+r}$ its elements have the same value (or are identified by the same class) is called a **spatial run** at location s of length r.

Spatial run statistic

The total number of runs is defined as:

$$SR^Q = n + \sum_{s \in S} \sum_{j=1}^{n_s} I_j^s$$

where $I_j^s = 1\, if\ X_{s_j - 1} \neq X_{s_j}\ and 0\ otherwise$ for $j = 1, 2, ..., n_s$
Following result by the Central Limit Theorem, the asymtotical distribution of SR^Q is:

$$SR^Q = N(\mu_{SR^Q}, \sigma_{SR^Q})$$

In the one-tailed case, we must distinguish the lower-tailed test and the upper-tailed, which are associated with homogeneity and heterogeneity respectively. In the case of the lower-tailed test, the following hypotheses are used:

$H_0 : \{X_s\}_{s \in S}$ is i.i.d.
H_1: The spatial distribution of the values of the categorical variable is more homogeneous than under the null hypothesis (according to the fixed association scheme). In the upper-tailed test, the following hypotheses are used:

$H_0 : \{X_s\}_{s \in S}$ is i.i.d.
H_1: The spatial distribution of the values of the categorical variable is more heterogeneous than under the null hypothesis (according to the fixed association scheme).

These hypotheses provide a decision rule regarding the degree of homogeneity in the spatial distribution of the values of the spatial categorical random variable.

Control arguments

`seedinit`	Numerical value for the seed (only for boot version). Default value seedinit=123

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`
Manuel Ruiz	`<manuel.ruiz@upct.es>`

References

- Ruiz, M., López, F., and Páez, A. (2021). A test for global and local homogeneity of categorical data based on spatial runs. *Working paper*.

See Also

local.sp.runs.test, dgp.spq, Q.test,

Examples

```
# Case 1: SRQ test based on factor and knn

n <- 100
cx <- runif(n)
cy <- runif(n)
x <- cbind(cx,cy)
listw <- spdep::knearneigh(cbind(cx,cy), k=3)
p <- c(1/6,3/6,2/6)
rho <- 0.5
fx <- dgp.spq(listw = listw, p = p, rho = rho)
srq <- sp.runs.test(fx = fx, listw = listw)
print(srq)
plot(srq)

# Boots Version
control <- list(seedinit = 1255)
srq <- sp.runs.test(fx = fx, listw = listw, distr = "bootstrap" , nsim = 299, control = control)
print(srq)
plot(srq)

# Case 2: SRQ test with formula, a sf object (points) and knn
data("FastFood.sf")
x <- sf::st_coordinates(sf::st_centroid(FastFood.sf))
listw <- spdep::knearneigh(x, k=4)
formula <- ~ Type
srq <- sp.runs.test(formula = formula, data = FastFood.sf, listw = listw)
print(srq)
plot(srq)
# Version boots
srq <- sp.runs.test(formula = formula, data = FastFood.sf, listw = listw,
distr = "bootstrap", nsim = 199)
print(srq)
plot(srq)

# Case 3: SRQ test (permutation) using formula with a sf object (polygons) and nb
library(sf)
fname <- system.file("shape/nc.shp", package="sf")
nc <- sf::st_read(fname)
listw <- spdep::poly2nb(as(nc,"Spatial"), queen = FALSE)
p <- c(1/6,3/6,2/6)
rho = 0.5
co <- sf::st_coordinates(sf::st_centroid(nc))
```

```
nc$fx <- dgp.spq(listw = listw, p = p, rho = rho)
plot(nc["fx"])
formula <- ~ fx
srq <- sp.runs.test(formula = formula, data = nc, listw = listw,
distr = "bootstrap", nsim = 399)
print(srq)
plot(srq)

# Case 4: SRQ test (Asymptotic) using formula with a sf object (polygons) and nb
data(provinces_spain)
# sf::sf_use_s2(FALSE)
listw <- spdep::poly2nb(provinces_spain, queen = FALSE)
provinces_spain$Coast <- factor(provinces_spain$Coast)
levels(provinces_spain$Coast) = c("no","yes")
plot(provinces_spain["Coast"])
formula <- ~ Coast
srq <- sp.runs.test(formula = formula, data = provinces_spain, listw = listw)
print(srq)
plot(srq)

# Boots version
srq <- sp.runs.test(formula = formula, data = provinces_spain, listw = listw,
distr = "bootstrap", nsim = 299)
print(srq)
plot(srq)

# Case 5: SRQ test based on a distance matrix (inverse distance)
N <- 100
cx <- runif(N)
cy <- runif(N)
data <- as.data.frame(cbind(cx,cy))
data <- sf::st_as_sf(data,coords = c("cx","cy"))
n = dim(data)[1]
dis <- 1/matrix(as.numeric(sf::st_distance(data,data)),ncol=n,nrow=n)
diag(dis) <- 0
dis <- (dis < quantile(dis,.10))*dis
p <- c(1/6,3/6,2/6)
rho <- 0.5
fx <- dgp.spq(listw = dis , p = p, rho = rho)
srq <- sp.runs.test(fx = fx, listw = dis)
print(srq)
plot(srq)

srq <- sp.runs.test(fx = fx, listw = dis, data = data)
print(srq)
plot(srq)

# Boots version
srq <- sp.runs.test(fx = fx, listw = dis, data = data, distr = "bootstrap", nsim = 299)
print(srq)
plot(srq)

# Case 6: SRQ test based on a distance matrix (inverse distance)
```

```
data("FastFood.sf")
# sf::sf_use_s2(FALSE)
n = dim(FastFood.sf)[1]
dis <- 1000000/matrix(as.numeric(sf::st_distance(FastFood.sf,FastFood.sf)), ncol = n, nrow = n)
diag(dis) <- 0
dis <- (dis < quantile(dis,.005))*dis
p <- c(1/6,3/6,2/6)
rho = 0.5
co <- sf::st_coordinates(sf::st_centroid(FastFood.sf))
FastFood.sf$fx <- dgp.spq(p = p, listw = dis, rho = rho)
plot(FastFood.sf["fx"])
formula <- ~ fx

# Boots version
srq <- sp.runs.test(formula = formula, data = FastFood.sf, listw = dis,
distr = "bootstrap", nsim = 299)
print(srq)
plot(srq)
```

---

summary.spjctest *Summary of estimated objects of class* spjctest.

Description

This function summarizes estimated *spjctest* objects. The tables in the output include basic information for each test. blablabla...

Usage

```
## S3 method for class 'spjctest'
summary(object, ...)
```

Arguments

`object`	An *spjctest* object including a list of *htest*.
`...`	further arguments passed to or from other methods.

Value

An object of class *summary.spjctest*

Author(s)

Fernando López	`<fernando.lopez@upct.es>`
Román Mínguez	`<roman.minguez@uclm.es>`
Antonio Páez	`<paezha@gmail.com>`

Manuel Ruiz <manuel.ruiz@upct.es>

See Also

print.summary.spqtest, joincount.test, joincount.multi

Examples

```
## Multinomial + Binomial using a sf multipolygon
data("provinces_spain")
sf::sf_use_s2(FALSE)
provinces_spain$Mal2Fml <- factor(provinces_spain$Mal2Fml > 100)
levels(provinces_spain$Mal2Fml) = c("men","woman")
provinces_spain$Older <- cut(provinces_spain$Older, breaks = c(-Inf,19,22.5,Inf))
levels(provinces_spain$Older) = c("low","middle","high")
f1 <- ~ Older + Mal2Fml
jc1 <- jc.test(formula = f1,
               data = provinces_spain,
               distr = "mc",
               alternative = "greater",
               zero.policy = TRUE)
summary(jc1)
```

summary.spqtest — *Summary of estimated objects of class* spqtest.

Description

This function summarizes estimated *spqtest* objects. The tables in the output include basic information for each test. blablabla...

Usage

```
## S3 method for class 'spqtest'
summary(object, ...)
```

Arguments

object	An *spqtest* object including a list of *htest*.
...	further arguments passed to or from other methods.

Value

An object of class *summary.spqtest*

Author(s)

Fernando López	<fernando.lopez@upct.es>
Román Mínguez	<roman.minguez@uclm.es>
Antonio Páez	<paezha@gmail.com>
Manuel Ruiz	<manuel.ruiz@upct.es>

See Also

print.summary.spqtest

Examples

```
# Example 1: With coordinates
N <- 100
cx <- runif(N)
cy <- runif(N)
coor <- cbind(cx,cy)
p <- c(1/6,3/6,2/6)
rho = 0.5
listw <- spdep::nb2listw(spdep::knn2nb(
              spdep::knearneigh(cbind(cx, cy), k = 4)))
fx <- dgp.spq(list = listw, p = p, rho = rho)
q.test <- Q.test(fx = fx, coor = coor, m = 3, r = 1)
summary(q.test)
plot(q.test)
```

Index