Package 'spdep'

September 13, 2024

Version 1.3-6

Date 2024-08-31

Title Spatial Dependence: Weighting Schemes, Statistics

Encoding UTF-8

Depends R (>= 3.3.0), methods, spData (>= 2.3.1), sf

Imports stats, deldir, boot (>= 1.3-1), graphics, utils, grDevices, units, s2, e1071, sp (>= 1.0)

Suggests spatialreg (>= 1.2-1), Matrix, parallel, dbscan, RColorBrewer, lattice, xtable, foreign, igraph, RSpectra, knitr, classInt, tmap, spam, ggplot2, rmarkdown, tinytest, rgeoda

URL https://github.com/r-spatial/spdep/, https://r-spatial.github.io/spdep/

BugReports https://github.com/r-spatial/spdep/issues/

Description A collection of functions to create spatial weights matrix objects from polygon 'contiguities', from point patterns by distance and tessellations, for summarizing these objects, and for permitting their use in spatial data analysis, including regional aggregation by minimum spanning tree; a collection of tests for spatial 'autocorrelation', including global 'Morans I' and 'Gearys C' proposed by 'Cliff' and 'Ord' (1973, ISBN: 0850860369) and (1981, ISBN: 0850860814), 'Hubert/Mantel' general cross product statistic, Empirical Bayes estimates and 'Assunção/Reis' (1999) <doi:10.1002/(SICI)1097-0258(19990830)18:16%3C2147::AID-SIM179%3E3.0.CO;2-I> Index, 'Getis/Ord' G ('Getis' and 'Ord' 1992) <doi:10.1111/j.1538-4632.1992.tb00261.x> and multicoloured join count statistics, 'APLE' ('Li 'et al.') <doi:10.1111/j.1538-4632.2007.00708.x>, local 'Moran's I', 'Gearys C' ('Anselin' 1995) <doi:10.1111/j.1538-4632.1995.tb00338.x> and 'Getis/Ord' G ('Ord' and 'Getis' 1995) <doi:10.1111/j.1538-4632.1995.tb00912.x>, 'saddlepoint' approximations ('Tiefelsdorf' 2002) <doi:10.1111/j.1538-4632.2002.tb01084.x> and exact tests

for global and local 'Moran's I' ('Bivand et al.' 2009) <doi:10.1016/j.csda.2008.07.021> and 'LOSH' local indicators of spatial heteroscedasticity ('Ord' and 'Getis') <doi:10.1007/s00168-011-0492-y>. The implementation of most of these measures is described in 'Bivand' and 'Wong' (2018) <doi:10.1007/s11749-018-0599-x>, with further extensions in 'Bivand' (2022) <doi:10.1111/gean.12319>. 'Lagrange' multiplier tests for spatial dependence in linear models are provided ('Anselin et al'. 1996) <doi:10.1016/0166-0462(95)02111-6>, as are 'Rao' score tests for hypothesised spatial 'Durbin' models based on linear models ('Koley' and 'Bera' 2023) <doi:10.1080/17421772.2023.2256810>.
From 'spdep' and 'spatialreg' versions >= 1.2-1, the model fitting functions previously present in this package are defunct in 'spdep' and may be found in 'spatialreg'.

License GPL (>= 2)

VignetteBuilder knitr

RoxygenNote RoxygenNote: 6.1.1

NeedsCompilation yes

Author Roger Bivand [cre, aut] (<https://orcid.org/0000-0003-2392-6140>),
Micah Altman [ctb],
Luc Anselin [ctb],
Renato Assunção [ctb],
Anil Bera [ctb],
Olaf Berke [ctb],
F. Guillaume Blanchet [ctb],
Marilia Carvalho [ctb],
Bjarke Christensen [ctb],
Yongwan Chun [ctb],
Carsten Dormann [ctb],
Stéphane Dray [ctb],
Dewey Dunnington [ctb] (<https://orcid.org/0000-0002-9415-4582>),
Virgilio Gómez-Rubio [ctb],
Malabika Koley [ctb],
Tomasz Kossowski [ctb] (<https://orcid.org/0000-0002-9976-4398>),
Elias Krainski [ctb],
Pierre Legendre [ctb],
Nicholas Lewin-Koh [ctb],
Angela Li [ctb],
Giovanni Millo [ctb],
Werner Mueller [ctb],
Hisaji Ono [ctb],
Josiah Parry [ctb] (<https://orcid.org/0000-0001-9910-865X>),
Pedro Peres-Neto [ctb],
Michał Pietrzak [ctb] (<https://orcid.org/0000-0002-9263-4478>),
Gianfranco Piras [ctb],
Markus Reder [ctb],
Jeff Sauer [ctb],

Michael Tiefelsdorf [ctb],
René Westerholt [ctb],
Justyna Wilk [ctb] (<https://orcid.org/0000-0003-1495-2910>),
Levi Wolf [ctb],
Danlin Yu [ctb]

Maintainer Roger Bivand <Roger.Bivand@nhh.no>

Repository CRAN

Date/Publication 2024-09-13 14:00:02 UTC

Contents

`aggregate.nb` *Aggregate a spatial neighbours object*

Description

The method aggregates a spatial neighbours object, creating a new object listing the neighbours of the aggregates.

Usage

```
## S3 method for class 'nb'
aggregate(x, IDs, remove.self = TRUE, ...)
```

Arguments

`x`	an nb neighbour object
`IDs`	a character vector of IDs grouping the members of the neighbour object
`remove.self`	default TRUE: remove self-neighbours resulting from aggregation
`...`	unused - arguments passed through

Value

an nb neighbour object, with empty aggregates dropped.

Note

Method suggested by Roberto Patuelli

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
data(used.cars, package="spData")
data(state)
cont_st <- match(attr(usa48.nb, "region.id"), state.abb)
cents <- as.matrix(as.data.frame(state.center))[cont_st,]
opar <- par(mfrow=c(2,1))
plot(usa48.nb, cents, xlim=c(-125, -65), ylim=c(25, 50))
IDs <- as.character(state.division[cont_st])
agg_cents <- aggregate(cents, list(IDs), mean)
agg_nb <- aggregate(usa48.nb, IDs)
plot(agg_nb, agg_cents[, 2:3], xlim=c(-125, -65), ylim=c(25, 50))
text(agg_cents[, 2:3], agg_cents[, 1], cex=0.6)
par(opar)
```

---

airdist *Measure distance from plot*

Description

Measure a distance between two points on a plot using `locator`; the function checks `par("plt")` and `par("usr")` to try to ensure that the aspect ratio y/x is 1, that is that the units of measurement in both x and y are equivalent.

Usage

```
airdist(ann=FALSE)
```

Arguments

`ann`	annotate the plot with line measured and distance

Value

a list with members:

`dist`	distance measured
`coords`	coordinates between which distance is measured

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`locator`

`autocov_dist` *Distance-weighted autocovariate*

Description

Calculates the autocovariate to be used in autonormal, autopoisson or autologistic regression. Three distance-weighting schemes are available.

Usage

```
autocov_dist(z, xy, nbs = 1, type = "inverse", zero.policy = NULL,
 style = "B", longlat=NULL)
```

Arguments

`z`	the response variable
`xy`	a matrix of coordinates or a SpatialPoints, `sf` or `sfc` points object
`nbs`	neighbourhood radius; default is 1
`type`	the weighting scheme: "one" gives equal weight to all data points in the neighbourhood; "inverse" (the default) weights by inverse distance; "inverse.squared" weights by the square of "inverse"
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
`style`	default “B” (changed from “W” 2015-01-27); style can take values “W”, “B”, “C”, “U”, and “S”
`longlat`	TRUE if point coordinates are longitude-latitude decimal, in which case distances are measured in kilometers; if xy is a SpatialPoints object, the value is taken from the object itself

Value

A numeric vector of autocovariate values

Note

The validity of this approach strongly hinges on the correct choice of the neighbourhood scheme! Using ‘`style="B"`’ ensures symmetry of the neighbourhood matrix (i.e. $w_{nm} = w_{mn}$). Please see Bardos et al. (2015) for details.

Author(s)

Carsten F. Dormann and Roger Bivand

References

Augustin N.H., Mugglestone M.A. and Buckland S.T. (1996) An autologistic model for the spatial distribution of wildlife. *Journal of Applied Ecology*, 33, 339-347; Gumpertz M.L., Graham J.M. and Ristaino J.B. (1997) Autologistic model of spatial pattern of Phytophthora epidemic in bell pepper: effects of soil variables on disease presence. *Journal of Agricultural, Biological and Environmental Statistics*, 2, 131-156; Bardos, D.C., Guillera-Arroita, G. and Wintle, B.A. (2015) Valid auto-models for spatially autocorrelated occupancy and abundance data. arXiv, 1501.06529.

See Also

nb2listw

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#xy <- cbind(columbus$X, columbus$Y)
xy <- st_coordinates(st_centroid(st_geometry(columbus),
 of_largest_polygon=TRUE))
ac1a <- autocov_dist(columbus$CRIME, xy, nbs=10, style="B",
 type="one")
acinva <- autocov_dist(columbus$CRIME, xy, nbs=10, style="B",
 type="inverse")
acinv2a <- autocov_dist(columbus$CRIME, xy, nbs=10, style="B",
 type="inverse.squared")
plot(ac1a ~ columbus$CRIME, pch=16, ylim=c(0,9000))
points(acinva ~ columbus$CRIME, pch=16, col="red")
points(acinv2a ~ columbus$CRIME, pch=16, col="blue")
legend("topleft", legend=c("one", "inverse", "inverse.squared"),
 col=c("black", "red", "blue"), bty="n", pch=16)
nb <- dnearneigh(xy, 0, 10)
lw <- nb2listw(nb, style="B")
ac1b <- lag(lw, columbus$CRIME)
all.equal(ac1b, ac1a)
nbd <- nbdists(nb, xy)
gl <- lapply(nbd, function(x) 1/x)
lw <- nb2listw(nb, glist=gl, style="B")
acinvb <- lag(lw, columbus$CRIME)
all.equal(acinvb, acinva)
gl2 <- lapply(nbd, function(x) 1/(x^2))
lw <- nb2listw(nb, glist=gl2, style="B")
acinv2b <- lag(lw, columbus$CRIME)
all.equal(acinv2b, acinv2a)
#xy <- SpatialPoints(xy)
#acinva <- autocov_dist(columbus$CRIME, xy, nbs=10, style="W",
# type="inverse")
#nb <- dnearneigh(xy, 0, 10)
#nbd <- nbdists(nb, xy)
#gl <- lapply(nbd, function(x) 1/x)
#lw <- nb2listw(nb, glist=gl)
#acinvb <- lag(lw, columbus$CRIME)
#all.equal(acinvb, acinva)
```

```
acinvc <- autocov_dist(columbus$CRIME, st_centroid(st_geometry(columbus),
 of_largest_polygon=TRUE), nbs=10, style="W", type="inverse")
all.equal(acinvc, acinva)
```

---

bhicv *Data set with 4 life condition indices of Belo Horizonte region*

Description

The data are collected inthe Atlas of condition indices published by the Joao Pinheiro Foundation and UNDP.

Format

A shape polygon object with seven variables:

id The identificator

Name Name of city

Population The population of city

HLCI Health Life Condition Index

ELCI Education Life Condition Index

CLCI Children Life Condition Index

ELCI Economic Life Condition Index

Examples

```
(GDAL37 <- as.numeric_version(unname(sf_extSoftVersion()["GDAL"])) >= "3.7.0")
file <- "etc/shapes/bhicv.gpkg.zip"
zipfile <- system.file(file, package="spdep")
if (GDAL37) {
    bh <- st_read(zipfile)
} else {
    td <- tempdir()
    bn <- sub(".zip", "", basename(file), fixed=TRUE)
    target <- unzip(zipfile, files=bn, exdir=td)
    bh <- st_read(target)
}
```

card *Cardinalities for neighbours lists*

Description

The function tallies the numbers of neighbours of regions in the neighbours list.

Usage

```
card(nb)
```

Arguments

nb	a neighbours list object of class nb

Details

"nb" objects are stored as lists of integer vectors, where the vectors contain either the indices in the range `1:n` for `n` as `length(nb)` of the neighbours of region `i`, or `as.integer(0)` to signal no neighbours. The function `card(nb)` is used to extract the numbers of neighbours from the "nb" object.

Value

An integer vector of the numbers of neighbours of regions in the neighbours list.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Bivand R, Pebesma EJ, Gomez-Rubio V, (2008) *Applied Spatial Data Analysis with R*, Springer, New York, pp. 239-251; Bivand R, Portnov B, (2004) Exploring spatial data analysis techniques using R: the case of observations with no neighbours. In: Anselin L, Florax R, Rey S, (eds.), *Advances in Spatial Econometrics, Methodology, Tools and Applications*. Berlin: Springer-Verlag, pp. 121-142, doi:10.1007/9783662056172_6.

See Also

summary.nb

Examples

```
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
table(card(col.gal.nb))
```

cell2nb *Generate neighbours list for grid cells*

Description

The function generates a list of neighbours for a grid of cells. Helper functions are used to convert to and from the vector indices for row and column grid positions, and rook (shared edge) or queen (shared edge or vertex) neighbour definitions are applied by type. If torus is TRUE, the grid is mapped onto a torus, removing edge effects.

Usage

```
cell2nb(nrow, ncol, type="rook", torus=FALSE, legacy=FALSE, x=NULL)
vi2mrc(i, nrow, ncol)
```

Arguments

nrow	number of rows in the grid, may also be an object inheriting from class "SpatialGrid" or "GridTopology" only in cell2nb
ncol	number of columns in the grid; if nrow or x is an object inheriting from class "SpatialGrid" or "GridTopology", it may be omitted
type	default rook, may also be queen
torus	default FALSE, if TRUE map grid onto torus
legacy	default FALSE, nrow/ncol reversed, if TRUE wrong col/row directions (see https://github.com/r-spatial/spdep/issues/20)
x	if given, an object inheriting from class "SpatialGrid" or "GridTopology", and replaces nrow and ncol
i	vector of vector indices corresponding to rowcol, a matrix with two columns of row, column indices

Value

The function returns an object of class nb with a list of integer vectors containing neighbour region number ids. See card for details of "nb" objects.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb, card

Examples

```
nb7rt <- cell2nb(7, 7)
summary(nb7rt)
xyc <- attr(nb7rt, "region.id")
xy <- matrix(as.integer(unlist(strsplit(xyc, ":"))), ncol=2, byrow=TRUE)
plot(nb7rt, xy)
nb7rt <- cell2nb(7, 7, torus=TRUE)
summary(nb7rt)
run <- FALSE
if (require("sp", quietly=TRUE)) run <- TRUE
if (run) {
# https://github.com/r-spatial/spdep/issues/20
GT <- GridTopology(c(1, 1), c(1, 1), c(10, 50))
SPix <- as(SpatialGrid(GT), "SpatialPixels")
nb_rook_cont <- poly2nb(as(SPix, "SpatialPolygons"), queen=FALSE)
nb_rook_dist <- dnearneigh(coordinates(SPix), 0, 1.01)
all.equal(nb_rook_cont, nb_rook_dist, check.attributes=FALSE)
## [1] TRUE
}
if (run) {
t.nb <- cell2nb(GT, type='rook', legacy=TRUE)
isTRUE(all.equal(nb_rook_cont, t.nb, check.attributes=FALSE))
## [1] FALSE
}
if (run) {
t.nb <- cell2nb(GT, type='rook')
isTRUE(all.equal(nb_rook_cont, t.nb, check.attributes=FALSE))
## [1] TRUE
}
if (run) {
# https://github.com/r-spatial/spdep/issues/55
# problem reported in issue caused by rep() cycling in unexpected order
GT <- GridTopology(c(1, 1), c(1, 1), c(22, 11))
SPix <- as(SpatialGrid(GT), "SpatialPixels")
nb_rook_cont <- poly2nb(as(SPix, "SpatialPolygons"), queen=FALSE)
nb_rook_dist <- dnearneigh(coordinates(SPix), 0, 1.01)
all.equal(nb_rook_cont, nb_rook_dist, check.attributes=FALSE)
}
if (run) {
t.nb <- cell2nb(GT, type='rook', legacy=TRUE)
isTRUE(all.equal(nb_rook_cont, t.nb, check.attributes=FALSE))
## [1] FALSE
}
if (run) {
t.nb <- cell2nb(GT, type='rook', legacy=FALSE)
isTRUE(all.equal(nb_rook_cont, t.nb, check.attributes=FALSE))
## [1] TRUE
}
```

---

choynowski *Choynowski probability map values*

Description

Calculates Choynowski probability map values.

Usage

```
choynowski(n, x, row.names=NULL, tol = .Machine$double.eps^0.5, legacy=FALSE)
```

Arguments

n	a numeric vector of counts of cases
x	a numeric vector of populations at risk
row.names	row names passed through to output data frame
tol	accumulate values for observed counts >= expected until value less than tol
legacy	default FALSE using vectorised alternating side ppois version, if true use original version written from sources and iterating down to tol

Value

A data frame with columns:

pmap	Poisson probability map values: probablility of getting a more "extreme" count than actually observed, one-tailed with less than expected and more than expected folded together
type	logical: TRUE if observed count less than expected

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Choynowski, M (1959) Maps based on probabilities, Journal of the American Statistical Association, 54, 385–388; Cressie, N, Read, TRC (1985), Do sudden infant deaths come in clusters? Statistics and Decisions, Supplement Issue 2, 333–349; Bailey T, Gatrell A (1995) Interactive Spatial Data Analysis, Harlow: Longman, pp. 300–303.

See Also

probmap

Examples

```
auckland <- st_read(system.file("shapes/auckland.gpkg", package="spData")[1], quiet=TRUE)
auckland.nb <- poly2nb(auckland)
res <- choynowski(auckland$M77_85, 9*auckland$Und5_81)
resl <- choynowski(auckland$M77_85, 9*auckland$Und5_81, legacy=TRUE)
all.equal(res, resl)
rt <- sum(auckland$M77_85)/sum(9*auckland$Und5_81)
ch_ppois_pmap <- numeric(length(auckland$Und5_81))
```

```
side <- c("greater", "less")
for (i in seq(along=ch_ppois_pmap)) {
  ch_ppois_pmap[i] <- poisson.test(auckland$M77_85[i], r=rt,
    T=(9*auckland$Und5_81[i]), alternative=side[(res$type[i]+1)])$p.value
}
all.equal(ch_ppois_pmap, res$pmap)
res1 <- probmap(auckland$M77_85, 9*auckland$Und5_81)
table(abs(res$pmap - res1$pmap) < 0.00001, res$type)
lt005 <- (res$pmap < 0.05) & (res$type)
ge005 <- (res$pmap < 0.05) & (!res$type)
cols <- rep("nonsig", length(lt005))
cols[lt005] <- "low"
cols[ge005] <- "high"
auckland$cols <- factor(cols)
plot(auckland[,"cols"], main="Probability map")
```

---

columbus *Columbus OH spatial analysis data set*

Description

The data set is now part of the spData package

Usage

```
data(columbus)
```

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
```

---

diffnb *Differences between neighbours lists*

Description

The function finds differences between lists of neighbours, returning a nb neighbour list of those found

Usage

```
diffnb(x, y, verbose=NULL)
```

Arguments

x	an object of class nb
y	an object of class nb
verbose	default NULL, use global option value; report regions ids taken from object attribute "region.id" with differences

Value

A neighbours list with class nb

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
rn <- row.names(columbus)
knn1 <- knearneigh(coords, 1)
knn2 <- knearneigh(coords, 2)
nb1 <- knn2nb(knn1, row.names=rn)
nb2 <- knn2nb(knn2, row.names=rn)
diffs <- diffnb(nb2, nb1)
opar <- par(no.readonly=TRUE)
plot(st_geometry(columbus), border="grey", reset=FALSE,
 main="Plot of first (black) and second (red)\nnearest neighbours")
plot(nb1, coords, add=TRUE)
plot(diffs, coords, add=TRUE, col="red", lty=2)
par(opar)
```

---

dnearneigh *Neighbourhood contiguity by distance*

Description

The function identifies neighbours of region points by Euclidean distance in the metric of the points between lower (greater than or equal to (changed from version 1.1-7)) and upper (less than or equal to) bounds, or with longlat = TRUE, by Great Circle distance in kilometers. If x is an "sf" object and use_s2= is TRUE, spherical distances in km are used.

Usage

```
dnearneigh(x, d1, d2, row.names = NULL, longlat = NULL, bounds=c("GE", "LE"),
 use_kd_tree=TRUE, symtest=FALSE, use_s2=packageVersion("s2") > "1.0.7", k=200,
 dwithin=TRUE)
```

Arguments

x	matrix of point coordinates, an object inheriting from SpatialPoints or an "sf" or "sfc" object; if the "sf" or "sfc" object geometries are in geographical coordinates (use_s2=FALSE, sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE), **s2** will be used to find the neighbours because it will (we hope) use spatial indexing https://github.com/r-spatial/s2/issues/125 as opposed to the legacy method which uses brute-force (at present **s2** also uses brute-force)
d1	lower distance bound in the metric of the points if planar coordinates, in km if in geographical coordinates
d2	upper distance boundd in the metric of the points if planar coordinates, in km if in geographical coordinates
row.names	character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1, nrow(x))
longlat	TRUE if point coordinates are geographical longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself, and overrides this argument if not NULL
bounds	character vector of length 2, default c("GE", "LE"), (GE: greater than or equal to, LE: less than or equal to) that is the finite and closed interval [d1, d2], d1 <= x <= d2. The first element may also be "GT" (GT: greater than), the second "LT" (LT: less than) for finite, open intervals excluding the bounds; the first bound default was changed from "GT" to "GE" in release 1.1-7. When creating multiple distance bands, finite, half-open right-closed intervals may be used until the final interval to avoid overlapping on bounds: "GE", "LT", that is [d1, d2), d1 <= x < d2
use_kd_tree	default TRUE, if TRUE, use **dbscan** frNN if available (permitting 3D distances).
symtest	Default FALSE; before release 1.1-7, TRUE - run symmetry check on output object, costly with large numbers of points.
use_s2	default=packageVersion("s2") > "1.0.7", as of **s2** > 1.0-7, distance bound computations use spatial indexing so when sf::sf_use_s2() is TRUE, s2::s2_dwithin_matrix() will be used for distances on the sphere for "sf" or "sfc" objects if **s2** > 1.0-7.
k	default 200, the number of closest points to consider when searching when using s2::s2_closest_edges()
dwithin	default TRUE, if FALSE, use s2::s2_closest_edges(), both if use_s2=TRUE, sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE; s2::s2_dwithin_matrix() yields the same lists of neighbours as s2::s2_closest_edges() is k= is set correctly.

Value

The function returns a list of integer vectors giving the region id numbers for neighbours satisfying the distance criteria. See card for details of "nb" objects.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knearneigh, card

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
rn <- row.names(columbus)
k1 <- knn2nb(knearneigh(coords))
all.linked <- max(unlist(nbdists(k1, coords)))
col.nb.0.all <- dnearneigh(coords, 0, all.linked, row.names=rn)
summary(col.nb.0.all, coords)
opar <- par(no.readonly=TRUE)
plot(st_geometry(columbus), border="grey", reset=FALSE,
 main=paste("Distance based neighbours 0-",  format(all.linked), sep=""))
plot(col.nb.0.all, coords, add=TRUE)
par(opar)
(sfc_obj <- st_centroid(st_geometry(columbus)))
col.nb.0.all_sf <- dnearneigh(sfc_obj, 0, all.linked, row.names=rn)
all.equal(col.nb.0.all, col.nb.0.all_sf, check.attributes=FALSE)
data(state)
us48.fipsno <- read.geoda(system.file("etc/weights/us48.txt",
 package="spdep")[1])
if (as.numeric(paste(version$major, version$minor, sep="")) < 19) {
 m50.48 <- match(us48.fipsno$"State.name", state.name)
} else {
 m50.48 <- match(us48.fipsno$"State_name", state.name)
}
xy <- as.matrix(as.data.frame(state.center))[m50.48,]
llk1 <- knn2nb(knearneigh(xy, k=1, longlat=FALSE))
(all.linked <- max(unlist(nbdists(llk1, xy, longlat=FALSE))))
ll.nb <- dnearneigh(xy, 0, all.linked, longlat=FALSE)
summary(ll.nb, xy, longlat=TRUE, scale=0.5)
gck1 <- knn2nb(knearneigh(xy, k=1, longlat=TRUE))
(all.linked <- max(unlist(nbdists(gck1, xy, longlat=TRUE))))
gc.nb <- dnearneigh(xy, 0, all.linked, longlat=TRUE)
summary(gc.nb, xy, longlat=TRUE, scale=0.5)
plot(ll.nb, xy)
plot(diffnb(ll.nb, gc.nb), xy, add=TRUE, col="red", lty=2)
title(main="Differences Euclidean/Great Circle")

#xy1 <- SpatialPoints((as.data.frame(state.center))[m50.48,],
#  proj4string=CRS("+proj=longlat +ellps=GRS80"))
#gck1a <- knn2nb(knearneigh(xy1, k=1))
#(all.linked <- max(unlist(nbdists(gck1a, xy1))))
#gc.nb <- dnearneigh(xy1, 0, all.linked)
#summary(gc.nb, xy1, scale=0.5)

xy1 <- st_as_sf((as.data.frame(state.center))[m50.48,], coords=1:2,
  crs=st_crs("OGC:CRS84"))
old_use_s2 <- sf_use_s2()
sf_use_s2(TRUE)
```

```
gck1b <- knn2nb(knearneigh(xy1, k=1))
system.time(o <- nbdists(gck1b, xy1))
(all.linked <- max(unlist(o)))
# use s2 brute-force dwithin_matrix approach for s2 <= 1.0.7
system.time(gc.nb.dwithin <- dnearneigh(xy1, 0, all.linked, use_s2=TRUE, dwithin=TRUE))
summary(gc.nb, xy1, scale=0.5)
# use s2 closest_edges approach s2 > 1.0.7
if (packageVersion("s2") > "1.0.7") {
(system.time(gc.nb.closest <- dnearneigh(xy1, 0, all.linked, dwithin=FALSE)))
}
if (packageVersion("s2") > "1.0.7") {
system.time(gc.nb.dwithin <- dnearneigh(xy1, 0, all.linked, use_s2=TRUE, dwithin=TRUE))
}
if (packageVersion("s2") > "1.0.7") {
summary(gc.nb.dwithin, xy1, scale=0.5)
}
if (packageVersion("s2") > "1.0.7") {
summary(gc.nb.closest, xy1, scale=0.5)
}
# use legacy symmetric brute-force approach
system.time(gc.nb.legacy <- dnearneigh(xy1, 0, all.linked, use_s2=FALSE))
summary(gc.nb, xy1, scale=0.5)
if (packageVersion("s2") > "1.0.7") all.equal(gc.nb.closest, gc.nb.dwithin, check.attributes=FALSE)
# legacy is ellipsoidal, s2 spherical, so minor differences expected
if (packageVersion("s2") > "1.0.7") all.equal(gc.nb, gc.nb.closest, check.attributes=FALSE)
all.equal(gc.nb, gc.nb.dwithin, check.attributes=FALSE)
sf_use_s2(old_use_s2)
# example of reading points with readr::read_csv() yielding a tibble
load(system.file("etc/misc/coords.rda", package="spdep"))
class(coords)
k1 <- knn2nb(knearneigh(coords, k=1))
all.linked <- max(unlist(nbdists(k1, coords)))
dnearneigh(coords, 0, all.linked)
```

`droplinks` *Drop and add links in a neighbours list*

Description

`droplinks` drops links to and from or just to a region from a neighbours list. The example corresponds to Fingleton's Table 1, (1999) p. 6, for lattices 5 to 19. `addlinks1` adds links from a single region to specified regions.

Usage

```
droplinks(nb, drop, sym=TRUE)
addlinks1(nb, from, to, sym=TRUE)
```

Arguments

nb	a neighbours list object of class nb
drop	either a logical vector the length of nb, or a character vector of named regions corresponding to nb's region.id attribute, or an integer vector of region numbers
sym	TRUE for removal of both "row" and "column" links, FALSE for only "row" links; when adding links, inserts links to the from region from the to regions
from	single from region for adding links, either a character vector of length 1 of the named from region corresponding to nb's region.id attribute, or an integer vector of length 1 holding a region number
to	to regions, either a character vector of named from regions corresponding to nb's region.id attribute, or an integer vector of region numbers

Value

The function returns an object of class nb with a list of integer vectors containing neighbour region number ids.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

B. Fingleton (1999) Spurious spatial regression: some Monte Carlo results with a spatial unit root and spatial cointegration, Journal of Regional Science 39, pp. 1–19.

See Also

is.symmetric.nb

Examples

```
rho <- c(0.2, 0.5, 0.95, 0.999, 1.0)
ns <- c(5, 7, 9, 11, 13, 15, 17, 19)
mns <- matrix(0, nrow=length(ns), ncol=length(rho))
rownames(mns) <- ns
colnames(mns) <- rho
mxs <- matrix(0, nrow=length(ns), ncol=length(rho))
rownames(mxs) <- ns
colnames(mxs) <- rho
for (i in 1:length(ns)) {
  nblist <- cell2nb(ns[i], ns[i])
  nbdropped <- droplinks(nblist, ((ns[i]*ns[i])+1)/2, sym=FALSE)
  listw <- nb2listw(nbdropped, style="W", zero.policy=TRUE)
  wmat <- listw2mat(listw)
  for (j in 1:length(rho)) {
    mat <- diag(ns[i]*ns[i]) - rho[j] * wmat
    res <- diag(solve(t(mat) %*% mat))
    mns[i,j] <- mean(res)
```

```
    mxs[i,j] <- max(res)
  }
}
print(mns)
print(mxs)
```

EBest *Global Empirical Bayes estimator*

Description

The function computes global empirical Bayes estimates for rates "shrunk" to the overall mean.

Usage

```
EBest(n, x, family="poisson")
```

Arguments

n	a numeric vector of counts of cases
x	a numeric vector of populations at risk
family	either "poisson" for rare conditions or "binomial" for non-rare conditions

Details

Details of the implementation for the "poisson" family are to be found in Marshall, p. 284–5, and Bailey and Gatrell p. 303–306 and exercise 8.2, pp. 328–330. For the "binomial" family, see Martuzzi and Elliott (implementation by Olaf Berke).

Value

A data frame with two columns:

raw	a numerical vector of raw (crude) rates
estmm	a numerical vector of empirical Bayes estimates

and a parameters attribute list with components:

a	global method of moments phi value
m	global method of moments gamma value

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> and Olaf Berke, Population Medicine, OVC, University of Guelph, CANADA

References

Marshall R M (1991) Mapping disease and mortality rates using Empirical Bayes Estimators, Applied Statistics, 40, 283–294; Bailey T, Gatrell A (1995) Interactive Spatial Data Analysis, Harlow: Longman, pp. 303–306, Martuzzi M, Elliott P (1996) Empirical Bayes estimation of small area prevalence of non-rare conditions, Statistics in Medicine 15, 1867–1873.

See Also

EBlocal, probmap, EBImoran.mc

Examples

```
auckland <- st_read(system.file("shapes/auckland.gpkg", package="spData")[1], quiet=TRUE)
res <- EBest(auckland$M77_85, 9*auckland$Und5_81)
attr(res, "parameters")
auckland$estmm000 <- res$estmm*1000
plot(auckland[,"estmm000"], breaks=c(0,2,2.5,3,3.5,5),
 main="Infant mortality per 1000 per year")
data(huddersfield, package="spData")
res <- EBest(huddersfield$cases, huddersfield$total, family="binomial")
round(res[,1:2],4)*100
```

EBImoran.mc *Permutation test for empirical Bayes index*

Description

An empirical Bayes index modification of Moran's I for testing for spatial autocorrelation in a rate, typically the number of observed cases in a population at risk. The index value is tested by using nsim random permutations of the index for the given spatial weighting scheme, to establish the rank of the observed statistic in relation to the nsim simulated values.

Usage

```
EBImoran.mc(n, x, listw, nsim, zero.policy = attr(listw, "zero.policy"),
 alternative = "greater", spChk=NULL, return_boot=FALSE,
 subtract_mean_in_numerator=TRUE)
```

Arguments

n	a numeric vector of counts of cases the same length as the neighbours list in listw
x	a numeric vector of populations at risk the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
nsim	number of permutations

zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "two.sided", or "less"
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
return_boot	return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest
subtract_mean_in_numerator	default TRUE, if TRUE subtract mean of z in numerator of EBI equation on p. 2157 in reference (consulted with Renato Assunção 2016-02-19); until February 2016 the default was FALSE agreeing with the printed paper.

Details

The statistic used is (m is the number of observations):

$$EBI = \frac{m}{\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij}} \frac{\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij} z_i z_j}{\sum_{i=1}^{m}(z_i - \bar{z})^2}$$

where:

$$z_i = \frac{p_i - b}{\sqrt{v_i}}$$

and:

$$p_i = n_i / x_i$$

$$v_i = a + (b/x_i)$$

$$b = \sum_{i=1}^{m} n_i / \sum_{i=1}^{m} x_i$$

$$a = s^2 - b/(\sum_{i=1}^{m} x_i/m)$$

$$s^2 = \sum_{i=1}^{m} x_i (p_i - b)^2 / \sum_{i=1}^{m} x_i$$

Value

A list with class htest and mc.sim containing the following components:

statistic	the value of the observed Moran's I.
parameter	the rank of the observed Moran's I.
p.value	the pseudo p-value of the test.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data, and the number of simulations.
res	nsim simulated values of statistic, final value is observed statistic
z	a numerical vector of Empirical Bayes indices as z above

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Assunção RM, Reis EA 1999 A new proposal to adjust Moran's I for population density. Statistics in Medicine 18, pp. 2147–2162; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x

See Also

moran, moran.mc, EBest

Examples

```
nc.sids <- st_read(system.file("shapes/sids.gpkg", package="spData")[1], quiet=TRUE)
rn <- as.character(nc.sids$FIPS)
ncCC89_nb <- read.gal(system.file("weights/ncCC89.gal", package="spData")[1],
 region.id=rn)
EBImoran.mc(nc.sids$SID74, nc.sids$BIR74,
 nb2listw(ncCC89_nb, style="B", zero.policy=TRUE), nsim=999,
 alternative="two.sided", zero.policy=TRUE)
sids.p <- nc.sids$SID74 / nc.sids$BIR74
moran.mc(sids.p, nb2listw(ncCC89_nb, style="B", zero.policy=TRUE),
 nsim=999, alternative="two.sided", zero.policy=TRUE)
```

EBlocal *Local Empirical Bayes estimator*

Description

The function computes local empirical Bayes estimates for rates "shrunk" to a neighbourhood mean for neighbourhoods given by the nb neighbourhood list.

Usage

```
EBlocal(ri, ni, nb, zero.policy = NULL, spChk = NULL, geoda=FALSE)
```

Arguments

ri	a numeric vector of counts of cases the same length as the neighbours list in nb; if there are many zero counts, some estimates may be affected by division by zero, see https://stat.ethz.ch/pipermail/r-sig-geo/2022-January/028882.html
ni	a numeric vector of populations at risk the same length as the neighbours list in nb
nb	a nb object of neighbour relationships

zero.policy	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
geoda	default=FALSE, following Marshall's algorithm as interpreted by Bailey and Gatrell, pp. 305-307, and exercise 8.2, pp. 328-330 for the definition of phi; TRUE for the definition of phi used in GeoDa (see discussion on OpenSpace mailing list June 2003: http://agec221.agecon.uiuc.edu/pipermail/openspace/2003-June/thread.html)

Details

Details of the implementation are to be found in Marshall, p. 286, and Bailey and Gatrell p. 307 and exercise 8.2, pp. 328–330. The example results do not fully correspond to the sources because of slightly differing neighbourhoods, but are generally close. If there are many zero counts, some estimates may be affected by division by zero, see https://stat.ethz.ch/pipermail/r-sig-geo/2022-January/028882.html.

Value

A data frame with two columns:

raw	a numerical vector of raw (crude) rates
est	a numerical vector of local empirical Bayes estimates

and a parameters attribute list with components (if both are zero, the estimate will be NaN, https://stat.ethz.ch/pipermail/r-sig-geo/2022-January/028882.html):

a	a numerical vector of local phi values
m	a numerical vector of local gamma values

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>, based on contributions by Marilia Carvalho

References

Marshall R M (1991) Mapping disease and mortality rates using Empirical Bayes Estimators, Applied Statistics, 40, 283–294; Bailey T, Gatrell A (1995) Interactive Spatial Data Analysis, Harlow: Longman, pp. 303–306.

See Also

EBest, probmap

Examples

```
auckland <- st_read(system.file("shapes/auckland.gpkg", package="spData")[1], quiet=TRUE)
auckland.nb <- poly2nb(auckland)
res <- EBlocal(auckland$M77_85,  9*auckland$Und5_81, auckland.nb)
auckland$est000 <- res$est*1000
plot(auckland[,"est000"], breaks=c(0,2,2.5,3,3.5,8),
 main="Infant mortality per 1000 per year")
```

`edit.nb` *Interactive editing of neighbours lists*

Description

The function provides simple interactive editing of neighbours lists to allow unneeded links to be deleted, and missing links to be inserted. It uses `identify` to pick the endpoints of the link to be deleted or added, and asks for confirmation before committing. If the result is not assigned to a new object, the editing will be lost - as in `edit`.

This method relies on direct contact with the graphics device. Do not use in RStudio.

Usage

```
## S3 method for class 'nb'
edit(name, coords, polys=NULL, ..., use_region.id=FALSE)
```

Arguments

`name`	an object of class `nb`
`coords`	matrix of region point coordinates; if missing and polys= inherits from `SpatialPolygons`, the label points of that object are used
`polys`	if polygon boundaries supplied, will be used as background; must inherit from `SpatialPolygons`
`...`	further arguments passed to or from other methods
`use_region.id`	default `FALSE`, in `identify` use 1-based observation numbers, otherwise use the `nb region.id` attribute values

Value

The function returns an object of class `nb` with the edited list of integer vectors containing neighbour region number ids, with added attributes tallying the added and deleted links.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`summary.nb`, `plot.nb`

Examples

```
## Not run:
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
class(columbus)
if (FALSE) nnb1 <- edit.nb(col.gal.nb, polys=as(columbus, "Spatial"))

## End(Not run)
```

`eire` *Eire data sets*

Description

The data set is now part of the spData package

Usage

```
data(eire)
```

`geary` *Compute Geary's C*

Description

A simple function to compute Geary's C, called by `geary.test` and `geary.mc`;

$$C = \frac{(n-1)}{2\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - x_j)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

`geary.intern` is an internal function used to vary the similarity criterion.

Usage

```
geary(x, listw, n, n1, S0, zero.policy=attr(listw, "zero.policy"), scale=TRUE)
```

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`
`n`	number of zones
`n1`	n - 1
`S0`	global sum of weights
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`scale`	default TRUE, may be FALSE to revert changes made to accommodate `localC` in November 2021 (see #151)

Value

a list with

C	Geary's C
K	sample kurtosis of x

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 17.

See Also

geary.test, geary.mc, sp.mantel.mc

Examples

```
data(oldcol)
col.W <- nb2listw(COL.nb, style="W")
str(geary(COL.OLD$CRIME, col.W, length(COL.nb), length(COL.nb)-1,
 Szero(col.W)))
```

---

geary.mc *Permutation test for Geary's C statistic*

Description

A permutation test for Geary's C statistic calculated by using nsim random permutations of x for the given spatial weighting scheme, to establish the rank of the observed statistic in relation to the nsim simulated values.

Usage

```
geary.mc(x, listw, nsim, zero.policy=attr(listw, "zero.policy"), alternative="greater",
 spChk=NULL, adjust.n=TRUE, return_boot=FALSE, na.action=na.fail, scale=TRUE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
nsim	number of permutations
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), or "less"; this reversal corresponds to that on geary.test described in the section on the output statistic value, based on Cliff and Ord 1973, p. 21 (changed 2011-04-11, thanks to Daniel Garavito).
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
return_boot	return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. na.pass is not permitted because it is meaningless in a permutation test.
scale	default TRUE, may be FALSE to revert changes made to accommodate localC in November 2021 (see #151)

Value

A list with class htest and mc.sim containing the following components:

statistic	the value of the observed Geary's C.
parameter	the rank of the observed Geary's C.
p.value	the pseudo p-value of the test.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data, and the number of simulations.
res	nsim simulated values of statistic, final value is observed statistic

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 63-5.

See Also

geary, geary.test

Examples

```
data(oldcol)
set.seed(1)
sim1 <- geary.mc(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 nsim=99, alternative="less")
sim1
mean(sim1$res)
var(sim1$res)
summary(sim1$res)
colold.lags <- nblag(COL.nb, 3)
sim2 <- geary.mc(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"), nsim=99)
sim2
summary(sim2$res)
sim3 <- geary.mc(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"), nsim=99)
sim3
summary(sim3$res)
crime <- COL.OLD$CRIME
is.na(crime) <- sample(1:length(crime), 10)
try(geary.mc(crime, nb2listw(COL.nb, style="W"), nsim=99,
 na.action=na.fail))
geary.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 na.action=na.omit)
geary.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 return_boot=TRUE, na.action=na.omit)
geary.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 na.action=na.exclude)
geary.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 return_boot=TRUE, na.action=na.exclude)
try(geary.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, na.action=na.pass))
```

`geary.test` *Geary's C test for spatial autocorrelation*

Description

Geary's test for spatial autocorrelation using a spatial weights matrix in weights list form. The assumptions underlying the test are sensitive to the form of the graph of neighbour relationships and other factors, and results may be checked against those of `geary.mc` permutations.

Usage

```
geary.test(x, listw, randomisation=TRUE, zero.policy=attr(listw, "zero.policy"),
    alternative="greater", spChk=NULL, adjust.n=TRUE, na.action=na.fail,
    scale=TRUE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
randomisation	variance of I calculated under the assumption of randomisation, if FALSE normality
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. na.pass is not permitted.
scale	default TRUE, may be FALSE to revert changes made to accommodate localC in November 2021 (see #151)

Value

A list with class htest containing the following components:

statistic	the value of the standard deviate of Geary's C, in the order given in Cliff and Ord 1973, p. 21, which is (EC - C) / sqrt(VC), that is with the sign reversed with respect to the more usual (C - EC) / sqrt(VC); this means that the "greater" alternative for the Geary C test corresponds to the "greater" alternative for Moran's I test.
p.value	the p-value of the test.
estimate	the value of the observed Geary's C, its expectation and variance under the method assumption.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the assumption used for calculating the standard deviate.
data.name	a character string giving the name(s) of the data.

Note

The derivation of the test (Cliff and Ord, 1981, p. 18) assumes that the weights matrix is symmetric. For inherently non-symmetric matrices, such as k-nearest neighbour matrices, listw2U() can be used to make the matrix symmetric. In non-symmetric weights matrix cases, the variance of the test statistic may be negative (thanks to Franz Munoz I for a well documented bug report). Geary's C is affected by non-symmetric weights under normality much more than Moran's I. From 0.4-35, the sign of the standard deviate of C is changed to match Cliff and Ord (1973, p. 21).

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 21, Cliff, A. D., Ord, J. K. 1973 Spatial Autocorrelation, Pion, pp. 15-16, 21; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s11749018-0599x

See Also

geary, geary.mc, listw2U

Examples

```
data(oldcol)
geary.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"))
geary.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
colold.lags <- nblag(COL.nb, 3)
geary.test(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"))
geary.test(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"), alternative="greater")
print(is.symmetric.nb(COL.nb))
coords.OLD <- cbind(COL.OLD$X, COL.OLD$Y)
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
geary.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"))
geary.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"),
 randomisation=FALSE)
cat("Note non-symmetric weights matrix - use listw2U()\n")
geary.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")))
geary.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")), randomisation=FALSE)
crime <- COL.OLD$CRIME
is.na(crime) <- sample(1:length(crime), 10)
try(geary.test(crime, nb2listw(COL.nb, style="W"), na.action=na.fail))
geary.test(crime, nb2listw(COL.nb, style="W"), zero.policy=TRUE,
 na.action=na.omit)
geary.test(crime, nb2listw(COL.nb, style="W"), zero.policy=TRUE,
 na.action=na.exclude)
try(geary.test(crime, nb2listw(COL.nb, style="W"), na.action=na.pass))
```

globalG.test *Global G test for spatial autocorrelation*

Description

The global G statistic for spatial autocorrelation, complementing the local Gi LISA measures: localG.

Usage

```
globalG.test(x, listw, zero.policy=attr(listw, "zero.policy"), alternative="greater",
 spChk=NULL, adjust.n=TRUE, B1correct=TRUE, adjust.x=TRUE, Arc_all_x=FALSE,
 na.action=na.fail)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw; if a sequence of distance bands is to be used, it is recommended that the weights style be binary (one of c("B", "C", "U")).
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
B1correct	default TRUE, if TRUE, the erratum referenced below: "On page 195, the coefficient of W2 in B1, (just below center of the page) should be 6, not 3." is applied; if FALSE, 3 is used (as in CrimeStat IV)
adjust.x	default TRUE, if TRUE, x values of observations with no neighbours are omitted in the denominator of G
Arc_all_x	default FALSE, if Arc_all_x=TRUE and adjust.x=TRUE, use the full x vector in part of the denominator term for G
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. na.pass is not permitted.

Value

A list with class `htest` containing the following components:

`statistic`	the value of the standard deviate of Getis-Ord G.
`p.value`	the p-value of the test.
`estimate`	the value of the observed statistic, its expectation and variance.
`alternative`	a character string describing the alternative hypothesis.
`data.name`	a character string giving the name(s) of the data.

Author(s)

Hisaji ONO <hi-ono@mn.xdsl.ne.jp> and Roger Bivand <Roger.Bivand@nhh.no>

References

Getis. A, Ord, J. K. 1992 The analysis of spatial association by use of distance statistics, *Geographical Analysis*, 24, p. 195; see also Getis. A, Ord, J. K. 1993 Erratum, *Geographical Analysis*, 25, p. 276; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x

See Also

localG

Examples

```
nc.sids <- st_read(system.file("shapes/sids.gpkg", package="spData")[1], quiet=TRUE)
sidsrate79 <- (1000*nc.sids$SID79)/nc.sids$BIR79
dists <- c(10, 20, 30, 33, 40, 50, 60, 70, 80, 90, 100)
ndists <- length(dists)
ZG <- vector(mode="list", length=ndists)
names(ZG) <- as.character(dists)
milesxy <- cbind(nc.sids$east, nc.sids$north)
for (i in 1:ndists) {
  thisnb <- dnearneigh(milesxy, 0, dists[i])
  thislw <- nb2listw(thisnb, style="B", zero.policy=TRUE)
  ZG[[i]] <- globalG.test(sidsrate79, thislw, zero.policy=TRUE)
}
t(sapply(ZG, function(x) c(x$estimate[1], x$statistic, p.value=unname(x$p.value))))
for (i in 1:ndists) {
  thisnb <- dnearneigh(milesxy, 0, dists[i])
  thislw <- nb2listw(thisnb, style="B", zero.policy=TRUE)
  ZG[[i]] <- globalG.test(sidsrate79, thislw, zero.policy=TRUE, alternative="two.sided")
}
t(sapply(ZG, function(x) c(x$estimate[1], x$statistic, p.value=unname(x$p.value))))
data(oldcol)
crime <- COL.OLD$CRIME
is.na(crime) <- sample(1:length(crime), 10)
res <- try(globalG.test(crime, nb2listw(COL.nb, style="B"),
 na.action=na.fail))
```

```
res
globalG.test(crime, nb2listw(COL.nb, style="B"), zero.policy=TRUE,
 na.action=na.omit)
globalG.test(crime, nb2listw(COL.nb, style="B"), zero.policy=TRUE,
 na.action=na.exclude)
try(globalG.test(crime, nb2listw(COL.nb, style="B"), na.action=na.pass))
```

Graph Components *Depth First Search on Neighbor Lists*

Description

`n.comp.nb()` finds the number of disjoint connected subgraphs in the graph depicted by `nb.obj` - a spatial neighbours list object.

Usage

```
n.comp.nb(nb.obj)
```

Arguments

`nb.obj`	a neighbours list object of class `nb`

Details

If `attr(nb.obj, "sym")` is `FALSE` and `igraph::components` is available, the components of the directed graph will be found by a simple breadth-first search; if `igraph::components` is not available, the object will be made symmetric (which may be time-consuming with large numbers of neighbours) and the components found by depth-first search. If `attr(nb.obj, "sym")` is `TRUE`, the components of the directed graph will be found by depth-first search. The time complexity of algorithms used in native code and through `igraph::components` is linear in the sum of the number of nodes and the number of edges in the graph, see https://github.com/r-spatial/spdep/issues/160 for details; very dense neighbour objects will have large numbers of edges.

Value

A list of:

`nc`	number of disjoint connected subgraphs
`comp.id`	vector with the indices of the disjoint connected subgraphs that the nodes in `nb.obj` belong to

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

`plot.nb`

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(st_geometry(columbus)))
plot(col.gal.nb, coords, col="grey")
col2 <- droplinks(col.gal.nb, 21)
res <- n.comp.nb(col2)
table(res$comp.id)
plot(col2, coords, add=TRUE)
points(coords, col=res$comp.id, pch=16)
run <- FALSE
if (require("igraph", quietly=TRUE) && require("spatialreg", quietly=TRUE)) run <- TRUE
if (run) {
B <- as(nb2listw(col2, style="B", zero.policy=TRUE), "CsparseMatrix")
g1 <- graph_from_adjacency_matrix(B, mode="undirected")
c1 <- components(g1)
print(c1$no == res$nc)
}
if (run) {
print(all.equal(c1$membership, res$comp.id))
}
if (run) {
print(all.equal(c1$csize, c(table(res$comp.id)), check.attributes=FALSE))
}
if (run) {
W <- as(nb2listw(col2, style="W", zero.policy=TRUE), "CsparseMatrix")
g1W <- graph_from_adjacency_matrix(W, mode="directed", weighted="W")
c1W <- components(g1W, mode="weak")
print(all.equal(c1W$membership, res$comp.id, check.attributes=FALSE))
}

if (run) {
data(house, package="spData")
house <- sf::st_as_sf(house)
k6 <- knn2nb(knearneigh(house, k=6))
is.symmetric.nb(k6)
}
if (run) {
print(k6)
}
if (run) {
length(k6) + sum(card(k6))
}
if (run) {
# no pre-computed graph components
str(attr(k6, "ncomp"))
}
if (run) {
# raising the subgraph compute ceiling to above |N|+|E| computes and stores the
# object in the neighbour object
set.SubgraphCeiling(180000L)
k6 <- knn2nb(knearneigh(house, k=6))
```

```
str(attr(k6, "ncomp"))
}
if (run) {
print(k6)
}
if (run) {
system.time(udir <- n.comp.nb(make.sym.nb(k6)))
}
if (run) {
system.time(dir <- n.comp.nb(k6))
}
if (run) {
udir$nc
}
if (run) {
dir$nc
}
if (run) {
all.equal(dir, udir)
}
```

---

graphneigh *Graph based spatial weights*

Description

Functions return a graph object containing a list with the vertex coordinates and the to and from indices defining the edges. Some/all of these functions assume that the coordinates are not exactly regularly spaced. The helper function graph2nb converts a graph object into a neighbour list. The plot functions plot the graph objects.

Usage

```
gabrielneigh(coords, nnmult=3)
relativeneigh(coords, nnmult=3)
soi.graph(tri.nb, coords, quadsegs=10)
graph2nb(gob, row.names=NULL,sym=FALSE)
## S3 method for class 'Gabriel'
plot(x, show.points=FALSE, add=FALSE, linecol=par(col), ...)
## S3 method for class 'relative'
plot(x, show.points=FALSE, add=FALSE, linecol=par(col),...)
```

Arguments

coords	matrix of region point coordinates or SpatialPoints object or sfc points object
nnmult	scaling factor for memory allocation, default 3; if higher values are required, the function will exit with an error; example below thanks to Dan Putler

tri.nb	a neighbor list created from tri2nb
quadsegs	number of line segments making a quarter circle buffer, see the nQuadSegs argument in geos_unary
gob	a graph object created from any of the graph funtions
row.names	character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1, nrow(x))
sym	a logical argument indicating whether or not neighbors should be symetric (if i->j then j->i)
x	object to be plotted
show.points	(logical) add points to plot
add	(logical) add to existing plot
linecol	edge plotting colour
...	further graphical parameters as in par(..)

Details

The graph functions produce graphs on a 2d point set that are all subgraphs of the Delaunay triangulation. The relative neighbor graph is defined by the relation, x and y are neighbors if

$$d(x, y) \leq min(max(d(x, z), d(y, z))|z \in S)$$

where d() is the distance, S is the set of points and z is an arbitrary point in S. The Gabriel graph is a subgraph of the delaunay triangulation and has the relative neighbor graph as a sub-graph. The relative neighbor graph is defined by the relation x and y are Gabriel neighbors if

$$d(x, y) \leq min((d(x, z)^2 + d(y, z)^2)^{1/2}|z \in S)$$

where x,y,z and S are as before. The sphere of influence graph is defined for a finite point set S, let r_x be the distance from point x to its nearest neighbor in S, and C_x is the circle centered on x. Then x and y are SOI neigbors iff C_x and C_y intersect in at least 2 places. From 2016-05-31, Computational Geometry in C code replaced by calls to functions in **dbscan** and **sf**; with a large quadsegs= argument, the behaviour of the function is the same, otherwise buffer intersections only closely approximate the original function.

See card for details of "nb" objects.

Value

A list of class Graph with the following elements

np	number of input points
from	array of origin ids
to	array of destination ids
nedges	number of edges in graph
x	input x coordinates
y	input y coordinates

The helper functions return an nb object with a list of integer vectors containing neighbour region number ids.

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

References

Matula, D. W. and Sokal R. R. 1980, Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane, Geographic Analysis, 12(3), pp. 205-222.

Toussaint, G. T. 1980, The relative neighborhood graph of a finite planar set, Pattern Recognition, 12(4), pp. 261-268.

Kirkpatrick, D. G. and Radke, J. D. 1985, A framework for computational morphology. In Computational Geometry, Ed. G. T. Toussaint, North Holland.

See Also

knearneigh, dnearneigh, knn2nb, card

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
sf_obj <- st_centroid(st_geometry(columbus), of_largest_polygon)
sp_obj <- as(sf_obj, "Spatial")
coords <- st_coordinates(sf_obj)
suppressMessages(col.tri.nb <- tri2nb(coords))
col.gab.nb <- graph2nb(gabrielneigh(coords), sym=TRUE)
col.rel.nb <- graph2nb(relativeneigh(coords), sym=TRUE)
par(mfrow=c(2,2))
plot(st_geometry(columbus), border="grey")
plot(col.tri.nb,coords,add=TRUE)
title(main="Delaunay Triangulation", cex.main=0.6)
plot(st_geometry(columbus), border="grey")
plot(col.gab.nb, coords, add=TRUE)
title(main="Gabriel Graph", cex.main=0.6)
plot(st_geometry(columbus), border="grey")
plot(col.rel.nb, coords, add=TRUE)
title(main="Relative Neighbor Graph", cex.main=0.6)
plot(st_geometry(columbus), border="grey")
if (require("dbscan", quietly=TRUE)) {
  col.soi.nb <- graph2nb(soi.graph(col.tri.nb,coords), sym=TRUE)
  plot(col.soi.nb, coords, add=TRUE)
  title(main="Sphere of Influence Graph", cex.main=0.6)
}
par(mfrow=c(1,1))
col.tri.nb_sf <- tri2nb(sf_obj)
all.equal(col.tri.nb, col.tri.nb_sf, check.attributes=FALSE)
col.tri.nb_sp <- tri2nb(sp_obj)
all.equal(col.tri.nb, col.tri.nb_sp, check.attributes=FALSE)
if (require("dbscan", quietly=TRUE)) {
  col.soi.nb_sf <- graph2nb(soi.graph(col.tri.nb, sf_obj), sym=TRUE)
  all.equal(col.soi.nb, col.soi.nb_sf, check.attributes=FALSE)
  col.soi.nb_sp <- graph2nb(soi.graph(col.tri.nb, sp_obj), sym=TRUE)
  all.equal(col.soi.nb, col.soi.nb_sp, check.attributes=FALSE)
```

```
}
col.gab.nb_sp <- graph2nb(gabrielneigh(sp_obj), sym=TRUE)
all.equal(col.gab.nb, col.gab.nb_sp, check.attributes=FALSE)
col.gab.nb_sf <- graph2nb(gabrielneigh(sf_obj), sym=TRUE)
all.equal(col.gab.nb, col.gab.nb_sf, check.attributes=FALSE)
col.rel.nb_sp <- graph2nb(relativeneigh(sp_obj), sym=TRUE)
all.equal(col.rel.nb, col.rel.nb_sp, check.attributes=FALSE)
col.rel.nb_sf <- graph2nb(relativeneigh(sf_obj), sym=TRUE)
all.equal(col.rel.nb, col.rel.nb_sf, check.attributes=FALSE)
dx <- rep(0.25*0:4,5)
dy <- c(rep(0,5),rep(0.25,5),rep(0.5,5), rep(0.75,5),rep(1,5))
m <- cbind(c(dx, dx, 3+dx, 3+dx), c(dy, 3+dy, dy, 3+dy))
cat(try(res <- gabrielneigh(m), silent=TRUE), "\n")
res <- gabrielneigh(m, nnmult=4)
summary(graph2nb(res))
grd <- as.matrix(expand.grid(x=1:5, y=1:5)) #gridded data
r2 <- gabrielneigh(grd)
set.seed(1)
grd1 <- as.matrix(expand.grid(x=1:5, y=1:5)) + matrix(runif(50, .0001, .0006), nrow=25)
r3 <- gabrielneigh(grd1)
opar <- par(mfrow=c(1,2))
plot(r2, show=TRUE, linecol=2)
plot(r3, show=TRUE, linecol=2)
par(opar)
# example of reading points with readr::read_csv() yielding a tibble
load(system.file("etc/misc/coords.rda", package="spdep"))
class(coords)
graph2nb(gabrielneigh(coords))
graph2nb(relativeneigh(coords))
```

`grid2nb` *Construct neighbours for a GridTopology*

Description

The function builds a neighbours list for a grid topology. It works for a k-dimentional grid topology, k>=1.

Usage

```
grid2nb(grid, d = grid@cells.dim,
        queen = TRUE, nb = TRUE, self = FALSE)
```

Arguments

`grid`	An object of class `GridTopology`. One can avoid to supply this by just suplying the dimentions in the `d` argument.
`d`	A scalar (for one dimentional grid) or a length k vector specyfying the number of grid cells in each direction of the k dimentions.

queen	Logical. Default is TRUE. To inform if the queen neighbourhood structure should be considered. If FALSE, only a hyper-cube with a common face will be considered neighbour. If TRUE, a single shared coordinate meets the contiguity condition.
nb	Default TRUE. If TRUE, return the result as a neighbours list with class nb. If FALSE, the result is a matrix with 3^k columns if self = TRUE or 3^k-1 if self = FALSE. Zeros are used for hyper-cubes at boundaries.
self	Default FALSE, to indicate if the hyper-cube neighbour itself should be considered a neighbour.

Value

Either a matrix, if "nb" is FALSE or a neighbours list with class nb. See card for details of "nb" objects.

Note

This applies to a k-dimentional grid topology.

Author(s)

Elias T Krainski <eliaskrainski@gmail.com>

See Also

poly2nb, summary.nb, card

Examples

```
nb <- grid2nb(d = c(5L, 5L, 5L))
nb
summary(nb)
if (require("sp", quietly=TRUE)) {
gt <- GridTopology(c(.125,.1), c(.25,.2), c(4L, 5L))
nb1 <- grid2nb(gt, queen = FALSE)
nb2 <- grid2nb(gt)

sg <- SpatialGrid(gt)
plot(sg, lwd=3)
plot(nb1, coordinates(sg), add=TRUE, lty=2, col=2, lwd=2)
plot(nb2, coordinates(sg), add=TRUE, lty=3, col=4)

str(grid2nb(d=5))
}
```

hotspot *Cluster Classifications for Local Indicators of Spatial Association and Local Indicators for Categorical Data*

Description

Used to return a factor showing so-called cluster classification for local indicators of spatial association for local Moran's I, local Geary's C (and its multivariate variant) and local Getis-Ord G. This factor vector can be added to a spatial object for mapping. When `obj` is of class `licd`, a list of up to six factors for measures of local composition (analytical and permutation), local configuration (analytical and permutation), and combined measures, both the interaction of composition and configuration, and a simplified recoding of these.

Usage

```
hotspot(obj, ...)

## Default S3 method:
hotspot(obj, ...)

## S3 method for class 'localmoran'
hotspot(obj, Prname, cutoff=0.005, quadrant.type="mean",
 p.adjust="fdr", droplevels=TRUE, ...)
## S3 method for class 'summary.localmoransad'
hotspot(obj, Prname, cutoff=0.005,
 quadrant.type="mean", p.adjust="fdr", droplevels=TRUE, ...)
## S3 method for class 'data.frame.localmoranex'
hotspot(obj, Prname, cutoff=0.005,
 quadrant.type="mean", p.adjust="fdr", droplevels=TRUE, ...)

## S3 method for class 'localG'
hotspot(obj, Prname, cutoff=0.005, p.adjust="fdr", droplevels=TRUE, ...)

## S3 method for class 'localC'
hotspot(obj, Prname, cutoff=0.005, p.adjust="fdr", droplevels=TRUE, ...)
## S3 method for class 'licd'
hotspot(obj, type = "both", cutoff = 0.05, p.adjust = "none",
 droplevels = TRUE, control = list(), ...)
```

Arguments

`obj`	An object of class `localmoran`, `localC` or `localG`
`Prname`	A character string, the name of the column containing the probability values to be classified by cluster type if found "interesting"
`cutoff`	Default 0.005, the probability value cutoff larger than which the observation is not found "interesting"

p.adjust	Default "fdr", the p.adjust() methood used, one of c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none")
droplevels	Default TRUE, should empty levels of the input cluster factor be dropped
quadrant.type	Default "mean", for "localmoran" objects only, can be c("mean", "median", "pysal") to partition the Moran scatterplot; "mean" partitions on the means of the variable and its spatial lag, "median" on medians of the variable and its spatial lag, "pysal" at zero for the centred variable and its spatial lag
type	When obj is of class licd, default both, may also be comp for local composition or config for local configuration
control	When obj is of class licd, default binomial_sidak 2, binomial_overlap TRUE, jcm_sidak 3. binomial_overlap may be set FALSE to avoid the Binomial probability values summing to more than unity - the tests in Boots (2003, p. 141) do overlap (>= and <=), and the Šidák exponents may be set to 1 to prevent by-observation correction for 2 Binomial and 3 Normal probability values per observation
...	other arguments passed to methods.

Value

A factor showing so-called cluster classification for local indicators of spatial association. When obj is of class licd, a list of up to six factors for measures of local composition (analytical and permutation), local configuration (analytical and permutation), and combined measures, both the interaction of composition and configuration, and a simplified recoding of these.

Author(s)

Roger Bivand

See Also

licd_multi

Examples

```
orig <- spData::africa.rook.nb
listw <- nb2listw(orig)
x <- spData::afcon$totcon

set.seed(1)
C <- localC_perm(x, listw)
Ch <- hotspot(C, Prname="Pr(z != E(Ci)) Sim", cutoff=0.05, p.adjust="none")
table(addNA(Ch))
set.seed(1)
I <- localmoran_perm(x, listw)
Ih <- hotspot(I, Prname="Pr(z != E(Ii)) Sim", cutoff=0.05, p.adjust="none")
table(addNA(Ih))
Is <- summary(localmoran.sad(lm(x ~ 1), nb=orig))
Ish <- hotspot(Is, Prname="Pr. (Sad)", cutoff=0.05, p.adjust="none")
table(addNA(Ish))
```

```
Ie <- as.data.frame(localmoran.exact(lm(x ~ 1), nb=orig))
Ieh <- hotspot(Ie, Prname="Pr. (exact)", cutoff=0.05, p.adjust="none")
table(addNA(Ieh))
set.seed(1)
G <- localG_perm(x, listw)
Gh <- hotspot(G, Prname="Pr(z != E(Gi)) Sim", cutoff=0.05, p.adjust="none")
table(addNA(Gh))
```

`include.self` *Include self in neighbours list*

Description

The function includes the region itself in its own list of neighbours, and sets attribute "self.included" to TRUE; `remove.self` reverts the effects of `include.self`.

Usage

```
include.self(nb)
remove.self(nb)
```

Arguments

nb	input neighbours list of class nb

Value

The function returns an object of class nb with a list of integer vectors containing neighbour region number ids; attribute "self.included" is set to TRUE.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
summary(col.gal.nb, coords)
summary(include.self(col.gal.nb), coords)
summary(remove.self(include.self(col.gal.nb)), coords)
```

`is.symmetric.nb` *Test a neighbours list for symmetry*

Description

Checks a neighbours list for symmetry/transitivity (if i is a neighbour of j, then j is a neighbour of i). This holds for distance and contiguity based neighbours, but not for k-nearest neighbours. The helper function `sym.attr.nb()` calls `is.symmetric.nb()` to set the `sym` attribute if needed, and `make.sym.nb` makes a non-symmetric list symmetric by adding neighbors. `is.symmetric.glist` checks a list of general weights corresponding to neighbours for symmetry for symmetric neighbours.

Usage

```
is.symmetric.nb(nb, verbose = NULL, force = FALSE)
sym.attr.nb(nb)
make.sym.nb(nb)
old.make.sym.nb(nb)
is.symmetric.glist(nb, glist)
```

Arguments

`nb`	an object of class `nb` with a list of integer vectors containing neighbour region number ids.
`verbose`	default NULL, use global option value; if TRUE prints non-matching pairs
`force`	do not respect a neighbours list `sym` attribute and test anyway
`glist`	list of general weights corresponding to neighbours

Value

TRUE if symmetric, FALSE if not; is.symmetric.glist returns a value with an attribute, "d", indicating for failed symmetry the largest failing value.

Note

A new version of `make.sym.nb` by Bjarke Christensen is now included. The older version has been renamed `old.make.sym.nb`, and their comparison constitutes a nice demonstration of vectorising speedup using `sapply` and `lapply` rather than loops. When any no-neighbour observations are present, `old.make.sym.nb` is used.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`read.gal`

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
ind <- row.names(as(columbus, "Spatial"))
print(is.symmetric.nb(col.gal.nb, verbose=TRUE, force=TRUE))
k4 <- knn2nb(knearneigh(coords, k=4), row.names=ind)
k4 <- sym.attr.nb(k4)
print(is.symmetric.nb(k4))
k4.sym <- make.sym.nb(k4)
print(is.symmetric.nb(k4.sym))
```

---

joincount.mc *Permutation test for same colour join count statistics*

Description

A permutation test for same colour join count statistics calculated by using nsim random permutations of fx for the given spatial weighting scheme, to establish the ranks of the observed statistics (for each colour) in relation to the nsim simulated values.

Usage

```
joincount.mc(fx, listw, nsim, zero.policy=attr(listw, "zero.policy"),
 alternative="greater", spChk=NULL)
```

Arguments

fx	a factor of the same length as the neighbours and weights objects in listw
listw	a listw object created for example by nb2listw
nsim	number of permutations
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "two.sided", or "less".
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()

Value

A list with class jclist of lists with class htest and mc.sim for each of the k colours containing the following components:

statistic	the value of the observed statistic.
parameter	the rank of the observed statistic.

method	a character string giving the method used.
data.name	a character string giving the name(s) of the data.
p.value	the pseudo p-value of the test.
alternative	a character string describing the alternative hypothesis.
estimate	the mean and variance of the simulated distribution.
res	nsim simulated values of statistic, the final element is the observed statistic

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 63-5.

See Also

joincount.test

Examples

```
data(oldcol)
HICRIME <- cut(COL.OLD$CRIME, breaks=c(0,35,80), labels=c("low","high"))
names(HICRIME) <- rownames(COL.OLD)
joincount.mc(HICRIME, nb2listw(COL.nb, style="B"), nsim=99, alternative="two.sided")
joincount.test(HICRIME, nb2listw(COL.nb, style="B"), alternative="two.sided")
```

---

joincount.multi *BB, BW and Jtot join count statistic for k-coloured factors*

Description

A function for tallying join counts between same-colour and different colour spatial objects, where neighbour relations are defined by a weights list. Given the global counts in each colour, expected counts and variances are calculated under non-free sampling, and a z-value reported. Since multiple tests are reported, no p-values are given, allowing the user to adjust the significance level applied. Jtot is the count of all different-colour joins.

Usage

```
joincount.multi(fx, listw, zero.policy = attr(listw, "zero.policy"),
 spChk = NULL, adjust.n=TRUE)
## S3 method for class 'jcmulti'
print(x, ...)
```

Arguments

fx	a factor of the same length as the neighbours and weights objects in listw
listw	a listw object created for example by nb2listw
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted consistently (up to and including spdep 0.3-28 the adjustment was inconsistent - thanks to Tomoki NAKAYA for a careful bug report)
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
x	object to be printed
...	arguments to be passed through for printing

Value

A matrix with class jcmulti with row and column names for observed and expected counts, variance, and z-value.

Note

The derivation of the test (Cliff and Ord, 1981, p. 18) assumes that the weights matrix is symmetric. For inherently non-symmetric matrices, such as k-nearest neighbour matrices, listw2U() can be used to make the matrix symmetric. In non-symmetric weights matrix cases, the variance of the test statistic may be negative.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

joincount.test

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
HICRIME <- cut(columbus$CRIME, breaks=c(0,35,80), labels=c("low","high"))
(nb <- poly2nb(columbus))
lw <- nb2listw(nb, style="B")
joincount.multi(HICRIME, lw)
col_geoms <- st_geometry(columbus)
```

```
col_geoms[21] <- st_buffer(col_geoms[21], dist=-0.05)
st_geometry(columbus) <- col_geoms
(nb <- poly2nb(columbus))
lw <- nb2listw(nb, style="B", zero.policy=TRUE)
joincount.multi(HICRIME, lw)
## Not run:
data(oldcol)
HICRIME <- cut(COL.OLD$CRIME, breaks=c(0,35,80), labels=c("low","high"))
names(HICRIME) <- rownames(COL.OLD)
joincount.multi(HICRIME, nb2listw(COL.nb, style="B"))
data(hopkins, package="spData")
image(1:32, 1:32, hopkins[5:36,36:5], breaks=c(-0.5, 3.5, 20),
 col=c("white", "black"))
box()
hopkins.rook.nb <- cell2nb(32, 32, type="rook")
unlist(spweights.constants(nb2listw(hopkins.rook.nb, style="B")))
hopkins.queen.nb <- cell2nb(32, 32, type="queen")
hopkins.bishop.nb <- diffnb(hopkins.rook.nb, hopkins.queen.nb, verbose=FALSE)
hopkins4 <- hopkins[5:36,36:5]
hopkins4[which(hopkins4 > 3, arr.ind=TRUE)] <- 4
hopkins4.f <- factor(hopkins4)
table(hopkins4.f)
joincount.multi(hopkins4.f, nb2listw(hopkins.rook.nb, style="B"))
cat("replicates Upton & Fingleton table 3.4 (p. 166)\n")
joincount.multi(hopkins4.f, nb2listw(hopkins.bishop.nb, style="B"))
cat("replicates Upton & Fingleton table 3.6 (p. 168)\n")
joincount.multi(hopkins4.f, nb2listw(hopkins.queen.nb, style="B"))
cat("replicates Upton & Fingleton table 3.7 (p. 169)\n")

## End(Not run)
GDAL37 <- as.numeric_version(unname(sf_extSoftVersion()["GDAL"])) >= "3.7.0"
file <- "etc/shapes/GB_2024_southcoast_50m.gpkg.zip"
zipfile <- system.file(file, package="spdep")
if (GDAL37) {
    sc50m <- st_read(zipfile)
} else {
    td <- tempdir()
    bn <- sub(".zip", "", basename(file), fixed=TRUE)
    target <- unzip(zipfile, files=bn, exdir=td)
    sc50m <- st_read(target)
}
sc50m$Winner <- factor(sc50m$Winner, levels=c("Con", "Green", "Lab", "LD"))
plot(sc50m[,"Winner"], pal=c("#2297E6", "#61D04F", "#DF536B", "#F5C710"))
nb_sc_50m <- poly2nb(sc50m, row.names=as.character(sc50m$Constituency))
sub2 <- attr(nb_sc_50m, "region.id")[attr(nb_sc_50m, "ncomp")$comp.id == 2L]
iowe <- match(sub2[1], attr(nb_sc_50m, "region.id"))
diowe <- c(st_distance(sc50m[iowe,], sc50m))
meet_criterion <- sum(diowe <= units::set_units(5000, "m"))
cands <- attr(nb_sc_50m, "region.id")[order(diowe)[1:meet_criterion]]
nb_sc_50m_iowe <- addlinks1(nb_sc_50m, from = cands[1],
 to = cands[3:meet_criterion])
ioww <- match(sub2[2], attr(nb_sc_50m, "region.id"))
dioww <- c(st_distance(sc50m[ioww,], sc50m))
```

```
meet_criterion <- sum(dioww <= units::set_units(5000, "m"))
cands <- attr(nb_sc_50m, "region.id")[order(dioww)[1:meet_criterion]]
nb_sc_50m_iow <- addlinks1(nb_sc_50m_iowe, from = cands[2], to = cands[3:meet_criterion])
nb_sc_1_2 <- nblag_cumul(nblag(nb_sc_50m_iow, 2))
joincount.multi(sc50m$Winner, nb2listw(nb_sc_1_2, style="B"))
```

`joincount.test` *BB join count statistic for k-coloured factors*

Description

The BB join count test for spatial autocorrelation using a spatial weights matrix in weights list form for testing whether same-colour joins occur more frequently than would be expected if the zones were labelled in a spatially random way. The assumptions underlying the test are sensitive to the form of the graph of neighbour relationships and other factors, and results may be checked against those of `joincount.mc` permutations.

Usage

```
joincount.test(fx, listw, zero.policy=attr(listw, "zero.policy"), alternative="greater",
 sampling="nonfree", spChk=NULL, adjust.n=TRUE)
## S3 method for class 'jclist'
print(x, ...)
```

Arguments

`fx`	a factor of the same length as the neighbours and weights objects in listw
`listw`	a `listw` object created for example by `nb2listw`
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
`sampling`	default “nonfree”, may be “free”
`adjust.n`	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted consistently (up to and including spdep 0.3-28 the adjustment was inconsistent - thanks to Tomoki NAKAYA for a careful bug report)
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`x`	object to be printed
`...`	arguments to be passed through for printing

Value

A list with class `jclist` of lists with class `htest` for each of the k colours containing the following components:

`statistic`	the value of the standard deviate of the join count statistic.
`p.value`	the p-value of the test.
`estimate`	the value of the observed statistic, its expectation and variance under non-free sampling.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string giving the method used.
`data.name`	a character string giving the name(s) of the data.

Note

The derivation of the test (Cliff and Ord, 1981, p. 18) assumes that the weights matrix is symmetric. For inherently non-symmetric matrices, such as k-nearest neighbour matrices, `listw2U()` can be used to make the matrix symmetric. In non-symmetric weights matrix cases, the variance of the test statistic may be negative.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, pp. 19-20.

See Also

`joincount.mc`, `joincount.multi`, `listw2U`

Examples

```
data(oldcol)
HICRIME <- cut(COL.OLD$CRIME, breaks=c(0,35,80), labels=c("low","high"))
names(HICRIME) <- rownames(COL.OLD)
joincount.test(HICRIME, nb2listw(COL.nb, style="B"))
joincount.test(HICRIME, nb2listw(COL.nb, style="B"), sampling="free")
joincount.test(HICRIME, nb2listw(COL.nb, style="C"))
joincount.test(HICRIME, nb2listw(COL.nb, style="S"))
joincount.test(HICRIME, nb2listw(COL.nb, style="W"))
by(card(COL.nb), HICRIME, summary)
print(is.symmetric.nb(COL.nb))
coords.OLD <- cbind(COL.OLD$X, COL.OLD$Y)
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
joincount.test(HICRIME, nb2listw(COL.k4.nb, style="B"))
cat("Note non-symmetric weights matrix - use listw2U()\n")
joincount.test(HICRIME, listw2U(nb2listw(COL.k4.nb, style="B")))
```

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
HICRIME <- cut(columbus$CRIME, breaks=c(0,35,80), labels=c("low","high"))
(nb <- poly2nb(columbus))
lw <- nb2listw(nb, style="B")
joincount.test(HICRIME, lw)
col_geoms <- st_geometry(columbus)
col_geoms[21] <- st_buffer(col_geoms[21], dist=-0.05)
st_geometry(columbus) <- col_geoms
(nb <- poly2nb(columbus))
lw <- nb2listw(nb, style="B", zero.policy=TRUE)
joincount.test(HICRIME, lw)
```

---

knearneigh *K nearest neighbours for spatial weights*

Description

The function returns a matrix with the indices of points belonging to the set of the k nearest neighbours of each other. If longlat = TRUE, Great Circle distances are used. A warning will be given if identical points are found.

Usage

```
knearneigh(x, k=1, longlat = NULL, use_kd_tree=TRUE)
```

Arguments

x	matrix of point coordinates, an object inheriting from SpatialPoints or an "sf" or "sfc" object; if the "sf" or "sfc" object geometries are in geographical coordinates (sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE), **s2** will be used to find the neighbours because it uses spatial indexing https://github.com/r-spatial/s2/issues/125 as opposed to the legacy method which uses brute-force
k	number of nearest neighbours to be returned; where identical points are present, k should be at least as large as the largest count of identical points (if k is smaller, an error will occur when s2 is used)
longlat	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself; longlat will override kd_tree
use_kd_tree	logical value, if the **dbscan** package is available, use for finding k nearest neighbours when longlat is FALSE, and when there are no identical points; from https://github.com/r-spatial/spdep/issues/38, the input data may have more than two columns if **dbscan** is used

Details

The underlying legacy C code is based on the knn function in the **class** package.

Value

A list of class knn

nn	integer matrix of region number ids
np	number of input points
k	input required k
dimension	number of columns of x
x	input coordinates

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knn, dnearneigh, knn2nb, kNN

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
col.knn <- knearneigh(coords, k=4)
plot(st_geometry(columbus), border="grey")
plot(knn2nb(col.knn), coords, add=TRUE)
title(main="K nearest neighbours, k = 4")
data(state)
us48.fipsno <- read.geoda(system.file("etc/weights/us48.txt",
 package="spdep")[1])
if (as.numeric(paste(version$major, version$minor, sep="")) < 19) {
 m50.48 <- match(us48.fipsno$"State.name", state.name)
} else {
 m50.48 <- match(us48.fipsno$"State_name", state.name)
}
xy <- as.matrix(as.data.frame(state.center))[m50.48,]
llk4.nb <- knn2nb(knearneigh(xy, k=4, longlat=FALSE))
gck4.nb <- knn2nb(knearneigh(xy, k=4, longlat=TRUE))
plot(llk4.nb, xy)
plot(diffnb(llk4.nb, gck4.nb), xy, add=TRUE, col="red", lty=2)
title(main="Differences between Euclidean and Great Circle k=4 neighbours")
summary(llk4.nb, xy, longlat=TRUE, scale=0.5)
summary(gck4.nb, xy, longlat=TRUE, scale=0.5)

#xy1 <- SpatialPoints((as.data.frame(state.center))[m50.48,],
#  proj4string=CRS("+proj=longlat +ellps=GRS80"))
#gck4a.nb <- knn2nb(knearneigh(xy1, k=4))
#summary(gck4a.nb, xy1, scale=0.5)

xy1 <- st_as_sf((as.data.frame(state.center))[m50.48,], coords=1:2,
  crs=st_crs("OGC:CRS84"))
old_use_s2 <- sf_use_s2()
```

```
sf_use_s2(TRUE)
system.time(gck4a.nb <- knn2nb(knearneigh(xy1, k=4)))
summary(gck4a.nb, xy1, scale=0.5)
sf_use_s2(FALSE)
system.time(gck4a.nb <- knn2nb(knearneigh(xy1, k=4)))
summary(gck4a.nb, xy1, scale=0.5)
sf_use_s2(old_use_s2)

# https://github.com/r-spatial/spdep/issues/38
if (require("dbscan", quietly=TRUE)) {
  set.seed(1)
  x <- cbind(runif(50), runif(50), runif(50))
  out <- knearneigh(x, k=5)
  knn2nb(out)
  try(out <- knearneigh(rbind(x, x[1:10,]), k=5))
}
```

---

knn2nb *Neighbours list from knn object*

Description

The function converts a knn object returned by knearneigh into a neighbours list of class nb with a list of integer vectors containing neighbour region number ids.

Usage

```
knn2nb(knn, row.names = NULL, sym = FALSE)
```

Arguments

knn	A knn object returned by knearneigh
row.names	character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1, nrow(x))
sym	force the output neighbours list to symmetry

Value

The function returns an object of class nb with a list of integer vectors containing neighbour region number ids. See card for details of "nb" objects.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knearneigh, card

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
coords <- st_coordinates(st_centroid(columbus))
col.knn <- knearneigh(coords, k=4)
plot(st_geometry(columbus), border="grey")
plot(knn2nb(col.knn), coords, add=TRUE)
title(main="K nearest neighbours, k = 4")
# example of reading points with readr::read_csv() yielding a tibble
load(system.file("etc/misc/coords.rda", package="spdep"))
class(coords)
knn2nb(knearneigh(coords, k=4))
```

`lag.listw` *Spatial lag of a numeric vector*

Description

Using a `listw` sparse representation of a spatial weights matrix, compute the lag vector Vx

Usage

```
## S3 method for class 'listw'
lag(x, var, zero.policy=attr(listw, "zero.policy"), NAOK=FALSE, ...)
```

Arguments

`x`	a `listw` object created for example by `nb2listw`
`var`	a numeric vector the same length as the neighbours list in listw
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`NAOK`	If 'FALSE', the presence of 'NA' values is regarded as an error; if 'TRUE' then any 'NA' or 'NaN' or 'Inf' values in var are represented as an NA lagged value.
`...`	additional arguments

Value

a numeric vector the same length as var

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

nb2listw

Examples

```
data(oldcol)
Vx <- lag.listw(nb2listw(COL.nb, style="W"), COL.OLD$CRIME)
plot(Vx, COL.OLD$CRIME)
plot(ecdf(COL.OLD$CRIME))
plot(ecdf(Vx), add=TRUE, col.points="red", col.hor="red")
is.na(COL.OLD$CRIME[5]) <- TRUE
VxNA <- lag.listw(nb2listw(COL.nb, style="W"), COL.OLD$CRIME, NAOK=TRUE)
```

---

lee *Compute Lee's statistic*

Description

A simple function to compute Lee's L statistic for bivariate spatial data;

$$L(x,y) = \frac{n}{\sum_{i=1}^{n}(\sum_{j=1}^{n} w_{ij})^2} \frac{\sum_{i=1}^{n}(\sum_{j=1}^{n} w_{ij}(x_i - \bar{x}))((\sum_{j=1}^{n} w_{ij}(y_j - \bar{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Usage

```
lee(x, y, listw, n, S2, zero.policy=attr(listw, "zero.policy"), NAOK=FALSE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
y	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
n	number of zones
S2	Sum of squared sum of weights by rows.
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
NAOK	if 'TRUE' then any 'NA' or 'NaN' or 'Inf' values in x are passed on to the foreign function. If 'FALSE', the presence of 'NA' or 'NaN' or 'Inf' values is regarded as an error.

Value

a list of

L	Lee's L statistic
local L	Lee's local L statistic

Author(s)

Roger Bivand and Virgiio GÃ³mez-Rubio <Virgilio.Gomez@uclm.es>

References

Lee (2001). Developing a bivariate spatial association measure: An integration of Pearson's r and Moran's I. J Geograph Syst 3: 369-385

See Also

lee.mc

Examples

```
data(boston, package="spData")
lw<-nb2listw(boston.soi)

x<-boston.c$CMEDV
y<-boston.c$CRIM
z<-boston.c$RAD

Lxy<-lee(x, y, lw, length(x), zero.policy=TRUE)
Lxz<-lee(x, z, lw, length(x), zero.policy=TRUE)
```

---

lee.mc *Permutation test for Lee's L statistic*

Description

A permutation test for Lee's L statistic calculated by using nsim random permutations of x and y for the given spatial weighting scheme, to establish the rank of the observed statistic in relation to the nsim simulated values.

Usage

```
lee.mc(x, y, listw, nsim, zero.policy=attr(listw, "zero.policy"), alternative="greater",
 na.action=na.fail, spChk=NULL, return_boot=FALSE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
y	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
nsim	number of permutations
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "two.sided", or "less".

na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. na.pass is not permitted because it is meaningless in a permutation test.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
return_boot	return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest

Value

A list with class htest and mc.sim containing the following components:

statistic	the value of the observed Lee's L.
parameter	the rank of the observed Lee's L.
p.value	the pseudo p-value of the test.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data, and the number of simulations.
res	nsim simulated values of statistic, final value is observed statistic

Author(s)

Roger Bivand, Virgilio GÃ³mez-Rubio <Virgilio.Gomez@uclm.es>

References

Lee (2001). Developing a bivariate spatial association measure: An integration of Pearson's r and Moran's I. J Geograph Syst 3: 369-385

See Also

lee

Examples

```
data(boston, package="spData")
lw<-nb2listw(boston.soi)

x<-boston.c$CMEDV
y<-boston.c$CRIM

lee.mc(x, y, nsim=99, lw, zero.policy=TRUE, alternative="two.sided")

#Test with missing values
x[1:5]<-NA
```

```
y[3:7]<-NA

lee.mc(x, y, nsim=99, lw, zero.policy=TRUE, alternative="two.sided",
   na.action=na.omit)
```

`lee.test` *Lee's L test for spatial autocorrelation*

Description

Lee's L test for spatial autocorrelation using a spatial weights matrix in weights list form. The assumptions underlying the test are sensitive to the form of the graph of neighbour relationships and other factors, and results may be checked against those of `lee.mc` permutations.

Usage

```
lee.test(x, y, listw, zero.policy=attr(listw, "zero.policy"),
 alternative="greater", na.action=na.fail, spChk=NULL)
```

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`y`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
`na.action`	a function (default `na.fail`), can also be `na.omit` or `na.exclude` - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted. If `na.pass` is used, zero is substituted for NA values in calculating the spatial lag
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`

Value

A list with class `htest` containing the following components:

`statistic`	the value of the standard deviate of Lee's L.
`p.value`	the p-value of the test.

estimate	the value of the observed Lee's L, its expectation and variance under the method assumption.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the assumption used for calculating the standard deviate.
data.name	a character string giving the name(s) of the data.

Note

See Lee (2004) for details on how the asymptotic expectation and variance of Lee's L is computed. In particular, check Lee (2004), table 1, page 1690.

This test may fail for large datasets as the computation of the asymptotic expectation and variance requires the use of dense matrices.

Author(s)

Roger Bivand and Virgilio GÃ³mez-Rubio <Virgilio.Gomez@uclm.es>

References

Lee (2004). A generalized significance testing method for global measures of spatial association: an extension of the Mantel test. Environment and Planning A 2004, volume 36, pages 1687 - 1703

See Also

lee, lee.mc, listw2U

Examples

```
data(oldcol)
col.W <- nb2listw(COL.nb, style="W")
crime <- COL.OLD$CRIME

lee.test(crime, crime, col.W, zero.policy=TRUE)

#Test with missing values
x<-crime
y<-crime
x[1:5]<-NA
y[3:7]<-NA

lee.test(x, y, col.W, zero.policy=TRUE, na.action=na.omit)
#  lee.test(x, y, col.W, zero.policy=TRUE)#Stops with an error



data(boston, package="spData")
lw<-nb2listw(boston.soi)

x<-boston.c$CMEDV
```

```
y<-boston.c$CRIM

lee.test(x, y, lw, zero.policy=TRUE, alternative="less")

#Test with missing values
x[1:5]<-NA
y[3:7]<-NA

lee.test(x, y, lw, zero.policy=TRUE, alternative="less", na.action=na.omit)
```

`licd_multi` *Local Indicators for Categorical Data*

Description

Local indicators for categorical data combine a measure of local composition in a window given by the per-observation set of neighbouring observations, with a local multi-category joincount test simplified to neighbours with the same or different categories compared to the focal observation

Usage

```
licd_multi(fx, listw, zero.policy = attr(listw, "zero.policy"), adjust.n = TRUE,
 nsim = 0L, iseed = NULL, no_repeat_in_row = FALSE, control = list())
```

Arguments

`fx`	a factor with two or more categories, of the same length as the neighbours and weights objects in listw
`listw`	a `listw` object created for example by `nb2listw`
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`adjust.n`	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

nsim	default 0, number of conditonal permutation simulations
iseed	default NULL, used to set the seed; the output will only be reproducible if the count of CPU cores across which computation is distributed is the same
no_repeat_in_row	default FALSE, if TRUE, sample conditionally in each row without replacements to avoid duplicate values, https://github.com/r-spatial/spdep/issues/124
control	comp_binary=TRUE, binomial_punif_alternative="greater", jcm_same_punif_alternative="less", jcm_diff_punif_alternative="greater"

Details

The original code may be found at doi:10.5281/zenodo.4283766

Value

local_comp	data.frame object with LICD local composition columns: ID, category_i, count_like_i, prop_i, count_nbs_i, pbinom_like_BW, pbinom_unlike_BW, pbinom_unlike_BW_alt, chi_BW_i, chi_K_i, anscombe_BW
local_config	data.frame object with LICD local configuration columns: ID, jcm_chi_obs, jcm_count_BB_obs, jcm_count_BW_obs, jcm_count_WW_obs, pval_jcm_obs_BB, pval_jcm_obs_WW, pval_jcm_obs_BW
local_comp_sim	data.frame object with permutation-based LICD local composition columns: ID, pbinom_like_BW, pbinom_unlike_BW, pbinom_unlike_BW_alt, rank_sim_chi_BW, rank_sim_chi_K, rank_sim_anscombe
local_config_sim	data.frame object with permutation-based LICD local configuration columns: ID, jcm_chi_sim_rank, pval_jcm_obs_BB_sim, pval_jcm_obs_BW_sim, pval_jcm_obs_WW_sim

Note

In order to increase the numbers of neighbours using nblag and nblag_cumul is advisable; use of binary weights is advised and are in any case used for the composition measure

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> based on earlier code by Tomasz M. Kossowski, Justyna Wilk and Michał B. Pietrzak

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 20;

Upton, G., Fingleton, B. 1985 Spatial data analysis by example: point pattern and qualitative data, Wiley, pp. 158–170;

Boots, B., 2003. Developing local measures of spatial association for categorical data. Journal of Geographical Systems 5, 139–160;

Boots, B., 2006. Local configuration measures for categorical spatial data: binary regular lattices. Journal of Geographical Systems 8 (1), 1–24;

Pietrzak, M.B., Wilk, J., Kossowski, T., Bivand, R.S., 2014. The application of local indicators for categorical data (LICD) in the spatial analysis of economic development. Comparative Economic Research 17 (4), 203–220 doi:10.2478/cer20140041;

Bivand, R.S., Wilk, J., Kossowski, T., 2017. Spatial association of population pyramids across Europe: The application of symbolic data, cluster analysis and join-count tests. Spatial Statistics 21 (B), 339–361 doi:10.1016/j.spasta.2017.03.003;

Francesco Carrer, Tomasz M. Kossowski, Justyna Wilk, Michał B. Pietrzak, Roger S. Bivand, The application of Local Indicators for Categorical Data (LICD) to explore spatial dependence in archaeological spaces, Journal of Archaeological Science, 126, 2021, doi:10.1016/j.jas.2020.105306

See Also

joincount.multi

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
HICRIME <- cut(columbus$CRIME, breaks=c(0,35,80), labels=c("low","high"))
(nb <- poly2nb(columbus))
lw <- nb2listw(nblag_cumul(nblag(nb, 2)), style="B")
obj <- licd_multi(HICRIME, lw)
str(obj)
h_obj <- hotspot(obj)
str(h_obj)
table(h_obj$both_recode)
columbus$both <- h_obj$both_recode
plot(columbus[, "both"])
GDAL37 <- as.numeric_version(unname(sf_extSoftVersion()["GDAL"])) >= "3.7.0"
file <- "etc/shapes/GB_2024_southcoast_50m.gpkg.zip"
zipfile <- system.file(file, package="spdep")
if (GDAL37) {
    sc50m <- st_read(zipfile)
} else {
    td <- tempdir()
    bn <- sub(".zip", "", basename(file), fixed=TRUE)
    target <- unzip(zipfile, files=bn, exdir=td)
    sc50m <- st_read(target)
}
sc50m$Winner <- factor(sc50m$Winner, levels=c("Con", "Green", "Lab", "LD"))
plot(sc50m[,"Winner"], pal=c("#2297E6", "#61D04F", "#DF536B", "#F5C710"))
nb_sc_50m <- poly2nb(sc50m, row.names=as.character(sc50m$Constituency))
sub2 <- attr(nb_sc_50m, "region.id")[attr(nb_sc_50m, "ncomp")$comp.id == 2L]
iowe <- match(sub2[1], attr(nb_sc_50m, "region.id"))
diowe <- c(st_distance(sc50m[iowe,], sc50m))
meet_criterion <- sum(diowe <= units::set_units(5000, "m"))
cands <- attr(nb_sc_50m, "region.id")[order(diowe)[1:meet_criterion]]
nb_sc_50m_iowe <- addlinks1(nb_sc_50m, from = cands[1],
 to = cands[3:meet_criterion])
ioww <- match(sub2[2], attr(nb_sc_50m, "region.id"))
```

```
dioww <- c(st_distance(sc50m[ioww,], sc50m))
meet_criterion <- sum(dioww <= units::set_units(5000, "m"))
cands <- attr(nb_sc_50m, "region.id")[order(dioww)[1:meet_criterion]]
nb_sc_50m_iow <- addlinks1(nb_sc_50m_iowe, from = cands[2], to = cands[3:meet_criterion])
nb_sc_1_2 <- nblag_cumul(nblag(nb_sc_50m_iow, 2))
lw <- nb2listw(nb_sc_1_2, style="B")
licd_obj <- licd_multi(sc50m$Winner, lw)
h_obj <- hotspot(licd_obj)
sc50m$both <- h_obj$both_recode
plot(sc50m[, "both"])
ljc <- local_joincount_uni(factor(sc50m$Winner == "LD"), chosen="TRUE", lw)
sc50m$LD_pv <- ljc[, 2]
plot(sc50m[, "LD_pv"], breaks=c(0, 0.025, 0.05, 0.1, 0.5, 1))
```

`listw2sn` *Spatial neighbour sparse representation*

Description

The function makes a `"spatial neighbour"` object representation (similar to the S-PLUS spatial statistics module representation of a `"listw"` spatial weights object. `sn2listw()` is the inverse function to `listw2sn()`, creating a `"listw"` object from a `"spatial neighbour"` object.

Usage

```
listw2sn(listw)
sn2listw(sn, style = NULL, zero.policy = NULL, from_mat2listw=FALSE)
```

Arguments

`listw`	a `listw` object from for example `nb2listw`
`sn`	a `spatial.neighbour` object
`style`	default NULL, missing, set to "M" and warning given; if not "M", passed to `nb2listw` to re-build the object
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
`from_mat2listw`	default FALSE, set TRUE if called from `mat2listw`

Value

`listw2sn()`returns a data frame with three columns, and with class `spatial.neighbour`:

`from`	region number id for the start of the link (S-PLUS row.id)
`to`	region number id for the end of the link (S-PLUS col.id)
`weights`	weight for this link

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

nb2listw

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
col.listw <- nb2listw(col.gal.nb)
col.listw$neighbours[[1]]
col.listw$weights[[1]]
col.sn <- listw2sn(col.listw)
str(col.sn)
```

---

lm.LMtests *Rao's score (a.k.a Lagrange Multiplier) diagnostics for spatial dependence in linear models*

Description

The function reports the estimates of tests chosen among five statistics for testing for spatial dependence in linear models. The statistics are the simple RS test for error dependence ("RSerr"), the simple RS test for a missing spatially lagged dependent variable ("RSlag"), variants of these adjusted for the presence of the other ("adjRSerr" tests for error dependence in the possible presence of a missing lagged dependent variable, "adjRSlag" the other way round), and a portmanteau test ("SARMA", in fact "RSerr" + "adjRSlag"). Note: from spdep 1.3-2, the tests are re-named "RS" - Rao's score tests, rather than "LM" - Lagrange multiplier tests to match the naming of tests from the same family in SDM.RStests.

Usage

```
lm.RStests(model, listw, zero.policy=attr(listw, "zero.policy"), test="RSerr",
 spChk=NULL, naSubset=TRUE)
lm.LMtests(model, listw, zero.policy=attr(listw, "zero.policy"), test="LMerr",
 spChk=NULL, naSubset=TRUE)
## S3 method for class 'RStestlist'
print(x, ...)
## S3 method for class 'RStestlist'
summary(object, p.adjust.method="none", ...)
## S3 method for class 'RStestlist.summary'
print(x, digits=max(3, getOption("digits") - 2), ...)
```

Arguments

model	an object of class lm returned by lm, or optionally a vector of externally calculated residuals (run though na.omit if any NAs present) for use when only "RSerr" is chosen; weights and offsets should not be used in the lm object
listw	a listw object created for example by nb2listw, expected to be row-standardised (W-style)
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
test	a character vector of tests requested chosen from RSerr, RSlag, adjRSerr, adjRSlag, SARMA; test="all" computes all the tests.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
naSubset	default TRUE to subset listw object for omitted observations in model object (this is a change from earlier behaviour, when the model$na.action component was ignored, and the listw object had to be subsetted by hand)
x, object	object to be printed
p.adjust.method	a character string specifying the probability value adjustment (see p.adjust) for multiple tests, default "none"
digits	minimum number of significant digits to be used for most numbers
...	printing arguments to be passed through

Details

The two types of dependence are for spatial lag ρ and spatial error λ:

$$\mathbf{y} = \mathbf{X}\beta + \rho\mathbf{W}_{(\mathbf{1})}\mathbf{y} + \mathbf{u},$$

$$\mathbf{u} = \lambda\mathbf{W}_{(\mathbf{2})}\mathbf{u} + \mathbf{e}$$

where $\mathbf{e}$ is a well-behaved, uncorrelated error term. Tests for a missing spatially lagged dependent variable test that $\rho = 0$, tests for spatial autocorrelation of the error $\mathbf{u}$ test whether $\lambda = 0$. $\mathbf{W}$ is a spatial weights matrix; for the tests used here they are identical.

Value

A list of class RStestlist of htest objects, each with:

statistic	the value of the Rao's score (a.k.a Lagrange multiplier) test.
parameter	number of degrees of freedom
p.value	the p-value of the test.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> and Andrew Bernat

References

Anselin, L. 1988 Spatial econometrics: methods and models. (Dordrecht: Kluwer); Anselin, L., Bera, A. K., Florax, R. and Yoon, M. J. 1996 Simple diagnostic tests for spatial dependence. Regional Science and Urban Economics, 26, 77–104 doi:10.1016/01660462(95)021116; Malabika Koley (2024) Specification Testing under General Nesting Spatial Model, https://sites.google.com/view/malabikakoley/research.

See Also

lm, SD.RStests

Examples

```
data(oldcol)
oldcrime.lm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD)
summary(oldcrime.lm)
lw <- nb2listw(COL.nb)
res <- lm.RStests(oldcrime.lm, listw=lw, test="all")
summary(res)
if (require("spatialreg", quietly=TRUE)) {
  oldcrime.slx <- lmSLX(CRIME ~ HOVAL + INC, data = COL.OLD, listw=lw)
  summary(lm.RStests(oldcrime.slx, listw=lw, test=c("adjRSerr", "adjRSlag")))
}
```

---

lm.morantest *Moran's I test for residual spatial autocorrelation*

Description

Moran's I test for spatial autocorrelation in residuals from an estimated linear model (lm()).

Usage

```
lm.morantest(model, listw, zero.policy=attr(listw, "zero.policy"),
 alternative = "greater", spChk=NULL, resfun=weighted.residuals, naSubset=TRUE)
```

Arguments

model	an object of class lm returned by lm; weights may be specified in the lm fit, but offsets should not be used
listw	a listw object created for example by nb2listw
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
resfun	default: weighted.residuals; the function to be used to extract residuals from the lm object, may be residuals, weighted.residuals, rstandard, or rstudent
naSubset	default TRUE to subset listw object for omitted observations in model object (this is a change from earlier behaviour, when the model$na.action component was ignored, and the listw object had to be subsetted by hand)

Value

A list with class htest containing the following components:

statistic	the value of the standard deviate of Moran's I.
p.value	the p-value of the test.
estimate	the value of the observed Moran's I, its expectation and variance under the method assumption.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 203,

See Also

lm.LMtests, lm

Examples

```
data(oldcol)
oldcrime1.lm <- lm(CRIME ~ 1, data = COL.OLD)
oldcrime.lm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD)
lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="W"))
lm.LMtests(oldcrime.lm, nb2listw(COL.nb, style="W"))
lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="S"))
lm.morantest(oldcrime1.lm, nb2listw(COL.nb, style="W"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
oldcrime.wlm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD,
 weights = I(1/AREA_PL))
lm.morantest(oldcrime.wlm, nb2listw(COL.nb, style="W"),
 resfun=weighted.residuals)
```

```
lm.morantest(oldcrime.wlm, nb2listw(COL.nb, style="W"),
 resfun=rstudent)
```

`lm.morantest.exact` *Exact global Moran's I test*

Description

The function implements Tiefelsdorf's exact global Moran's I test.

Usage

```
lm.morantest.exact(model, listw, zero.policy = attr(listw, "zero.policy"),
 alternative = "greater", spChk = NULL, resfun = weighted.residuals,
 zero.tol = 1e-07, Omega=NULL, save.M=NULL, save.U=NULL, useTP=FALSE,
 truncErr=1e-6, zeroTreat=0.1)
## S3 method for class 'moranex'
print(x, ...)
```

Arguments

`model`	an object of class `lm` returned by `lm`; weights may be specified in the `lm` fit, but offsets should not be used
`listw`	a `listw` object created for example by `nb2listw`
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`resfun`	default: weighted.residuals; the function to be used to extract residuals from the `lm` object, may be `residuals`, `weighted.residuals`, `rstandard`, or `rstudent`
`zero.tol`	tolerance used to find eigenvalues close to absolute zero
`Omega`	A SAR process matrix may be passed in to test an alternative hypothesis, for example `Omega <- invIrW(listw, rho=0.1); Omega <- tcrossprod(Omega)`, `chol()` is taken internally
`save.M`	return the full M matrix for use in `spdep:::exactMoranAlt`
`save.U`	return the full U matrix for use in `spdep:::exactMoranAlt`
`useTP`	default FALSE, if TRUE, use truncation point in integration rather than upper=Inf, see Tiefelsdorf (2000), eq. 6.7, p.69
`truncErr`	when useTP=TRUE, pass truncation error to truncation point function
`zeroTreat`	when useTP=TRUE, pass zero adjustment to truncation point function
`x`	a moranex object
`...`	arguments to be passed through

Value

A list of class `moranex` with the following components:

`statistic`	the value of the exact standard deviate of global Moran's I.
`p.value`	the p-value of the test.
`estimate`	the value of the observed global Moran's I.
`method`	a character string giving the method used.
`alternative`	a character string describing the alternative hypothesis.
`gamma`	eigenvalues (excluding zero values)
`oType`	usually set to "E"
`data.name`	a character string giving the name(s) of the data.
`df`	degrees of freedom

Author(s)

Markus Reder and Roger Bivand

References

Roger Bivand, Werner G. Müller and Markus Reder (2009) "Power calculations for global and local Moran's I." *Computational Statistics & Data Analysis* 53, 2859-2872.

See Also

`lm.morantest.sad`

Examples

```
eire <- st_read(system.file("shapes/eire.gpkg", package="spData")[1])
row.names(eire) <- as.character(eire$names)
eire.nb <- poly2nb(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
lm.morantest.exact(e.lm, nb2listw(eire.nb))
lm.morantest.exact(e.lm, nb2listw(eire.nb), useTP=TRUE)
```

`lm.morantest.sad` *Saddlepoint approximation of global Moran's I test*

Description

The function implements Tiefelsdorf's application of the Saddlepoint approximation to global Moran's I's reference distribution.

Usage

```
lm.morantest.sad(model, listw, zero.policy=attr(listw, "zero.policy"),
  alternative="greater", spChk=NULL, resfun=weighted.residuals,
  tol=.Machine$double.eps^0.5, maxiter=1000, tol.bounds=0.0001,
  zero.tol = 1e-07, Omega=NULL, save.M=NULL, save.U=NULL)
## S3 method for class 'moransad'
print(x, ...)
## S3 method for class 'moransad'
summary(object, ...)
## S3 method for class 'summary.moransad'
print(x, ...)
```

Arguments

`model`	an object of class `lm` returned by `lm`; weights may be specified in the `lm` fit, but offsets should not be used
`listw`	a `listw` object created for example by `nb2listw`
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`resfun`	default: weighted.residuals; the function to be used to extract residuals from the `lm` object, may be `residuals`, `weighted.residuals`, `rstandard`, or `rstudent`
`tol`	the desired accuracy (convergence tolerance) for `uniroot`
`maxiter`	the maximum number of iterations for `uniroot`
`tol.bounds`	offset from bounds for `uniroot`
`zero.tol`	tolerance used to find eigenvalues close to absolute zero
`Omega`	A SAR process matrix may be passed in to test an alternative hypothesis, for example `Omega <- invIrW(listw, rho=0.1); Omega <- tcrossprod(Omega)`, `chol()` is taken internally
`save.M`	return the full M matrix for use in `spdep:::exactMoranAlt`

save.U	return the full U matrix for use in spdep:::exactMoranAlt
x	object to be printed
object	object to be summarised
...	arguments to be passed through

Details

The function involves finding the eigenvalues of an n by n matrix, and numerically finding the root for the Saddlepoint approximation, and should therefore only be used with care when n is large.

Value

A list of class moransad with the following components:

statistic	the value of the saddlepoint approximation of the standard deviate of global Moran's I.
p.value	the p-value of the test.
estimate	the value of the observed global Moran's I.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data.
internal1	Saddlepoint omega, r and u
internal2	f.root, iter and estim.prec from uniroot
df	degrees of freedom
tau	eigenvalues (excluding zero values)

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Tiefelsdorf, M. 2002 The Saddlepoint approximation of Moran's I and local Moran's Ii reference distributions and their numerical evaluation. Geographical Analysis, 34, pp. 187–206; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x

See Also

lm.morantest

Examples

```
eire <- st_read(system.file("shapes/eire.gpkg", package="spData")[1])
row.names(eire) <- as.character(eire$names)
eire.nb <- poly2nb(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
summary(lm.morantest.sad(e.lm, nb2listw(eire.nb)))
e.wlm <- lm(OWNCONS ~ ROADACC, data=eire, weights=RETSALE)
lm.morantest(e.wlm, nb2listw(eire.nb), resfun=rstudent)
lm.morantest.sad(e.wlm, nb2listw(eire.nb), resfun=rstudent)
```

---

localC *Compute Local Geary statistic*

Description

The Local Geary is a local adaptation of Geary's C statistic of spatial autocorrelation. The Local Geary uses squared differences to measure dissimilarity unlike the Local Moran. Low values of the Local Geary indicate positive spatial autocorrelation and large refers to negative spatial autocorrelation.

Inference for the Local Geary is based on a permutation approach which compares the observed value to the reference distribution under spatial randomness. `localC_perm()` returns a pseudo p-value. This is not an analytical p-value and is based on the number of permutations and as such should be used with care.

Usage

```
localC(x, ..., zero.policy=NULL)

## Default S3 method:
localC(x, listw, ..., zero.policy=attr(listw, "zero.policy"))

## S3 method for class 'formula'
localC(formula, data, listw, ..., zero.policy=attr(listw, "zero.policy"))

## S3 method for class 'list'
localC(x, listw, ..., zero.policy=attr(listw, "zero.policy"))

## S3 method for class 'matrix'
localC(x, listw, ..., zero.policy=attr(listw, "zero.policy"))

## S3 method for class 'data.frame'
localC(x, listw, ..., zero.policy=attr(listw, "zero.policy"))

localC_perm(x, ..., zero.policy=NULL, iseed=NULL, no_repeat_in_row=FALSE)
```

```
## Default S3 method:
localC_perm(x, listw, nsim = 499, alternative = "two.sided", ...,
 zero.policy=attr(listw, "zero.policy"), iseed=NULL, no_repeat_in_row=FALSE)

## S3 method for class 'formula'
localC_perm(formula, data, listw, nsim = 499,
 alternative = "two.sided", ..., zero.policy=attr(listw, "zero.policy"), iseed=NULL,
 no_repeat_in_row=FALSE)
```

Arguments

x	a numeric vector, numeric matrix, or list. See details for more.
formula	A one-sided formula determining which variables to be used.
listw	a listw object created for example by nb2listw.
data	Used when a formula is provided. A matrix or data frame containing the variables in the formula formula.
nsim	The number of simulations to be used for permutation test.
alternative	A character defining the alternative hypothesis. Must be one of "two.sided", "less" or "greater".
...	other arguments passed to methods.
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA.
iseed	default NULL, used to set the seed;the output will only be reproducible if the count of CPU cores across which computation is distributed is the same
no_repeat_in_row	default FALSE, if TRUE, sample conditionally in each row without replacements to avoid duplicate values, https://github.com/r-spatial/spdep/issues/124

Details

The Local Geary can be extended to a multivariate context. When x is a numeric vector, the univariate Local Geary will be calculated. To calculate the multivariate Local Moran provide either a list or a matrix. When x is a list, each element must be a numeric vector of the same length and of the same length as the neighbours in listw. In the case that x is a matrix the number of rows must be the same as the length of the neighbours in listw.

While not required in the univariate context, the standardized Local Geary is calculated. The multivariate Local Geary is *always* standardized.

The univariate Local Geary is calculated as $c_i = \sum_j w_{ij}(x_i - x_j)^2$ and the multivariate Local Geary is calculated as $c_{k,i} = \sum_{v=1}^{k} c_{v,i}$ as described in Anselin (2019).

Value

A numeric vector containing Local Geary statistic with attribute pseudo-p when localC_perm() is used. pseudo-p is an 8 column matrix containing

E.Ci	expectation of the Local Geary statistic based on permutation sample
Var.Ci	variance of Local Geary based on permutation sample
Z.Ci	standard deviate of Local Geary based on permutation sample
Pr()	p-value of Local Geary statistic using pnorm() using standard deviates based on permutation sample means and standard deviations
Pr() Sim	rank() and punif() of observed statistic rank for [0, 1] p-values using alternative=
Pr(folded) Sim	the simulation folded [0, 0.5] range ranked p-value (based on https://github.com/pysal/esda/blob/4a63e0b5df1e754b17b5f1205b8cadcbecc5e061/esda/crand.py#L211-L213)
Skewness	the output of e1071::skewness() for the permutation samples underlying the standard deviates
Kurtosis	the output of e1071::kurtosis() for the permutation samples underlying the standard deviates

Author(s)

Josiah Parry <josiah.parry@gmail.com> and Roger Bivand

References

Anselin, L. (1995), Local Indicators of Spatial Association—LISA. Geographical Analysis, 27: 93-115. doi:10.1111/j.15384632.1995.tb00338.x

Anselin, L. (2019), A Local Indicator of Multivariate Spatial Association: Extending Gearys c. Geogr Anal, 51: 133-150. doi:10.1111/gean.12164

Examples

```
orig <- spData::africa.rook.nb
listw <- nb2listw(orig)
x <- spData::afcon$totcon

(A <- localC(x, listw))
listw1 <- nb2listw(droplinks(sym.attr.nb(orig), 3, sym=TRUE), zero.policy=TRUE)
(A1 <- localC(x, listw1, zero.policy=FALSE))
(A2 <- localC(x, listw1, zero.policy=TRUE))
run <- FALSE
if (require(rgeoda, quietly=TRUE)) run <- TRUE
if (run) {
  W <- create_weights(as.numeric(length(x)))
  for (i in 1:length(listw$neighbours)) {
    set_neighbors_with_weights(W, i, listw$neighbours[[i]], listw$weights[[i]])
    update_weights(W)
  }
  set.seed(1)
  B <- local_geary(W, data.frame(x))
  all.equal(A, lisa_values(B))
}
if (run) {
  set.seed(1)
```

```
  C <- localC_perm(x, listw, nsim = 499, conditional=TRUE,
    alternative="two.sided")
  cor(ifelse(lisa_pvalues(B) < 0.5, lisa_pvalues(B), 1-lisa_pvalues(B)),
    attr(C, "pseudo-p")[,6])
}
# pseudo-p values probably wrongly folded https://github.com/GeoDaCenter/rgeoda/issues/28

tmap_ok <- FALSE
if (require(tmap, quietly=TRUE)) tmap_ok <- TRUE
if (run) {
  # doi: 10.1111/gean.12164
  guerry_path <- system.file("extdata", "Guerry.shp", package = "rgeoda")
  g <- st_read(guerry_path)[, 7:12]
  cor(st_drop_geometry(g)) #(Tab. 1)
  lw <- nb2listw(poly2nb(g))
  moran(g$Crm_prs, lw, n=nrow(g), S0=Szero(lw))$I
  moran(g$Crm_prp, lw, n=nrow(g), S0=Szero(lw))$I
  moran(g$Litercy, lw, n=nrow(g), S0=Szero(lw))$I
  moran(g$Donatns, lw, n=nrow(g), S0=Szero(lw))$I
  moran(g$Infants, lw, n=nrow(g), S0=Szero(lw))$I
  moran(g$Suicids, lw, n=nrow(g), S0=Szero(lw))$I
}
if (run) {
  o <- prcomp(st_drop_geometry(g), scale.=TRUE)
  cor(st_drop_geometry(g), o$x[,1:2])^2 #(Tab. 2)
}
if (run) {
  g$PC1 <- o$x[, "PC1"]
  brks <- c(min(g$PC1), natural_breaks(k=6, g["PC1"]), max(g$PC1))
  if (tmap_ok) {
    tmap4 <- packageVersion("tmap") >= "3.99"
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="PC1",
        fill.scale=tm_scale(values="brewer.rd_yl_gn", breaks=brks,
        midpoint=0), fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
        frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("PC1", breaks=brks, midpoint=0) +
      tm_borders() # Fig. 1
    }
  } else {
    pplot(g["PC1"], breaks=brks)
  }
}
if (run) {
  g$PC2 <- -1*o$x[, "PC2"] # eigenvalue sign arbitrary
  brks <- c(min(g$PC2), natural_breaks(k=6, g["PC2"]), max(g$PC2))
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="PC2",
        fill.scale=tm_scale(values="brewer.rd_yl_gn", breaks=brks,
        midpoint=0), fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
        frame=FALSE, item.r=0))
```

```
    } else {
      tm_shape(g) + tm_fill("PC2", breaks=brks, midpoint=0) +
      tm_borders() # Fig. 2
    }
  } else {
    plot(g["PC2"], breaks=brks)
  }
}
if (run) {
  w <- queen_weights(g)
  lm_PC1 <- local_moran(w, g["PC1"], significance_cutoff=0.01,
    permutations=99999)
  g$lm_PC1 <- factor(lisa_clusters(lm_PC1), levels=0:4,
    labels=lisa_labels(lm_PC1)[1:5])
  is.na(g$lm_PC1) <- g$lm_PC1 == "Not significant"
  g$lm_PC1 <- droplevels(g$lm_PC1)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lm_PC1",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lm_PC1", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 3
    }
  } else {
    plot(g["lm_PC1"])
  }
}
if (run) {
  set.seed(1)
  lm_PC1_spdep <- localmoran_perm(g$PC1, lw, nsim=9999)
  q <- attr(lm_PC1_spdep, "quadr")$pysal
  g$lm_PC1_spdep <- q
  is.na(g$lm_PC1_spdep) <- lm_PC1_spdep[,6] > 0.02 # note folded p-values
  g$lm_PC1_spdep <- droplevels(g$lm_PC1_spdep)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lm_PC1_spdep",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lm_PC1_spdep", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # rep. Fig. 3
    }
  } else {
    plot(g["lm_PC1_spdep"])
  }
}
```

```
if (run) {
  lg_PC1 <- local_g(w, g["PC1"], significance_cutoff=0.01,
    permutations=99999)
  g$lg_PC1 <- factor(lisa_clusters(lg_PC1), levels=0:2,
    labels=lisa_labels(lg_PC1)[0:3])
  is.na(g$lg_PC1) <- g$lg_PC1 == "Not significant"
  g$lg_PC1 <- droplevels(g$lg_PC1)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lg_PC1",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lg_PC1", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 4 (wrong)
    }
  } else {
    plot(g["lg_PC1"])
  }
  g$lg_PC1a <- cut(g$PC1, c(-Inf, mean(g$PC1), Inf), labels=c("Low", "High"))
  is.na(g$lg_PC1a) <- lisa_pvalues(lg_PC1) >= 0.01
  g$lg_PC1a <- droplevels(g$lg_PC1a)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lg_PC1a",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lg_PC1a", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 4 (guess)
    }
  } else {
    plot(g["lg_PC1"])
  }
}
if (run) {
  lc_PC1 <- local_geary(w, g["PC1"], significance_cutoff=0.01,
    permutations=99999)
  g$lc_PC1 <- factor(lisa_clusters(lc_PC1), levels=0:4,
    labels=lisa_labels(lc_PC1)[1:5])
  is.na(g$lc_PC1) <- g$lc_PC1 == "Not significant"
  g$lc_PC1 <- droplevels(g$lc_PC1)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc_PC1",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
```

```
    } else {
      tm_shape(g) + tm_fill("lc_PC1", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 5
    }
  } else {
    plot(g["lc_PC1"])
  }
}
if (run) {
  set.seed(1)
  system.time(lc_PC1_spdep <- localC_perm(g$PC1, lw, nsim=9999,
    alternative="two.sided"))
}
if (run) {
  if (require(parallel, quietly=TRUE)) {
    ncpus <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
# test with single core
    if (ncpus > 1L) ncpus <- 1L
    cores <- get.coresOption()
    set.coresOption(ncpus)
    system.time(lmc_PC1_spdep1 <- localC_perm(g$PC1, lw, nsim=9999,
      alternative="two.sided", iseed=1))
    set.coresOption(cores)
  }
}
if (run) {
  g$lc_PC1_spdep <- attr(lc_PC1_spdep, "cluster")
  is.na(g$lc_PC1_spdep) <- attr(lc_PC1_spdep, "pseudo-p")[,6] > 0.01
  g$lc_PC1_spdep <- droplevels(g$lc_PC1_spdep)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc_PC1_spdep",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lc_PC1_spdep", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # rep. Fig. 5
    }
  } else {
    plot(g["lc_PC1_spdep"])
  }
}
if (run) {
  g$both_PC1 <- interaction(g$lc_PC1, g$lm_PC1)
  g$both_PC1 <- droplevels(g$both_PC1)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="both_PC1",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
```

```
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("both_PC1", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 6
    }
  } else {
    plot(g["both_PC1"])
  }
}
if (run) {
  lc005_PC1 <- local_geary(w, g["PC1"], significance_cutoff=0.005,
    permutations=99999)
  g$lc005_PC1 <- factor(lisa_clusters(lc005_PC1), levels=0:4,
    labels=lisa_labels(lc005_PC1)[1:5])
  is.na(g$lc005_PC1) <- g$lc005_PC1 == "Not significant"
  g$lc005_PC1 <- droplevels(g$lc005_PC1)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc005_PC1",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lc005_PC1", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 7
    }
  } else {
    plot(g["lc005_PC1"])
}
if (run) {
  g$lc005_PC1_spdep <- attr(lc_PC1_spdep, "cluster")
  is.na(g$lc005_PC1_spdep) <- attr(lc_PC1_spdep, "pseudo-p")[,6] > 0.005
  g$lc005_PC1_spdep <- droplevels(g$lc005_PC1_spdep)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc005_PC1_spdep",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lc005_PC1_spdep", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # rep. Fig. 7
    }
  } else {
    plot(g["lc005_PC1_spdep"])
  }
}
if (run) {
  lc001_PC1 <- local_geary(w, g["PC1"], significance_cutoff=0.001,
    permutations=99999)
  g$lc001_PC1 <- factor(lisa_clusters(lc001_PC1), levels=0:4,
```

```
    labels=lisa_labels(lc001_PC1)[1:5])
  is.na(g$lc001_PC1) <- g$lc001_PC1 == "Not significant"
  g$lc001_PC1 <- droplevels(g$lc001_PC1)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc005_PC1",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lc001_PC1", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 8
    }
  } else {
    plot(g["lc001_PC1"])
  }
}
if (run) {
  g$lc001_PC1_spdep <- attr(lc_PC1_spdep, "cluster")
  is.na(g$lc001_PC1_spdep) <- attr(lc_PC1_spdep, "pseudo-p")[,6] > 0.001
  g$lc001_PC1_spdep <- droplevels(g$lc001_PC1_spdep)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc005_PC1_spdep",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lc001_PC1_spdep", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # rep. Fig. 8
    }
  } else {
    plot(g["lc001_PC1_spdep"])
  }
}
}
if (run) {
  lc_PC2 <- local_geary(w, g["PC2"], significance_cutoff=0.01,
    permutations=99999)
  g$lc_PC2 <- factor(lisa_clusters(lc_PC2), levels=0:4,
    labels=lisa_labels(lc_PC2)[1:5])
  is.na(g$lc_PC2) <- g$lc_PC2 == "Not significant"
  g$lc_PC2 <- droplevels(g$lc_PC2)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lc_PC2",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
```

```
      tm_shape(g) + tm_fill("lc_PC2", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 9
    }
  } else {
    plot(g["lc_PC2"])
  }
}
if (run) {
  lmc_PC <- local_multigeary(w, g[c("PC1","PC2")], significance_cutoff=0.00247,
    permutations=99999)
  g$lmc_PC <- factor(lisa_clusters(lmc_PC), levels=0:1,
    labels=lisa_labels(lmc_PC)[1:2])
  is.na(g$lmc_PC) <- g$lmc_PC == "Not significant"
  g$lmc_PC <- droplevels(g$lmc_PC)
  table(interaction((p.adjust(lisa_pvalues(lmc_PC), "fdr") < 0.01), g$lmc_PC))
}
if (run) {
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lmc_PC",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lmc_PC", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 10
    }
  } else {
    plot(g["lmc_PC"])
  }
}
if (run) {
  set.seed(1)
  lmc_PC_spdep <- localC_perm(g[c("PC1","PC2")], lw, nsim=9999, alternative="two.sided")
  all.equal(lisa_values(lmc_PC), c(lmc_PC_spdep))
}
if (run) {
  cor(attr(lmc_PC_spdep, "pseudo-p")[,6], lisa_pvalues(lmc_PC))
}
if (run) {
  g$lmc_PC_spdep <- attr(lmc_PC_spdep, "cluster")
  is.na(g$lmc_PC_spdep) <- p.adjust(attr(lmc_PC_spdep, "pseudo-p")[,6], "fdr") > 0.01
  g$lmc_PC_spdep <- droplevels(g$lmc_PC_spdep)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lmc_PC_spdep",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lmc_PC_spdep", textNA="Insignificant",
```

```
        colorNA="gray95") + tm_borders() # rep. Fig. 10
    }
  } else {
    plot(g["lmc_PC_spdep"])
  }
}
if (run) {
  lmc_vars <- local_multigeary(w, st_drop_geometry(g)[, 1:6],
    significance_cutoff=0.00247, permutations=99999)
  g$lmc_vars <- factor(lisa_clusters(lmc_vars), levels=0:1,
    labels=lisa_labels(lmc_vars)[1:2])
  is.na(g$lmc_vars) <- g$lmc_vars == "Not significant"
  g$lmc_vars <- droplevels(g$lmc_vars)
  table(interaction((p.adjust(lisa_pvalues(lmc_vars), "fdr") < 0.01),
    g$lmc_vars))
}
if (run) {
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lmc_vars",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lmc_vars", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # Fig. 11
    }
  } else {
    plot(g["lmc_vars"])
  }
}
if (run) {
  set.seed(1)
  system.time(lmc_vars_spdep <- localC_perm(st_drop_geometry(g)[, 1:6], lw,
    nsim=9999, alternative="two.sided"))
}
if (run) {
  all.equal(lisa_values(lmc_vars), c(lmc_vars_spdep))
}
if (run) {
  cor(attr(lmc_vars_spdep, "pseudo-p")[,6], lisa_pvalues(lmc_vars))
}
if (run) {
  if (require(parallel, quietly=TRUE)) {
    ncpus <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
# test with single core
    if (ncpus > 1L) ncpus <- 1L
    cores <- get.coresOption()
    set.coresOption(ncpus)
    system.time(lmc_vars_spdep1 <- localC_perm(st_drop_geometry(g)[, 1:6], lw,
      nsim=9999, alternative="two.sided", iseed=1))
    set.coresOption(cores)
```

```
  }
}
if (run) {
  all.equal(lisa_values(lmc_vars), c(lmc_vars_spdep1))
}
if (run) {
  cor(attr(lmc_vars_spdep1, "pseudo-p")[,6], lisa_pvalues(lmc_vars))
}
if (run) {
  g$lmc_vars_spdep <- attr(lmc_vars_spdep1, "cluster")
 is.na(g$lmc_vars_spdep) <- p.adjust(attr(lmc_vars_spdep1, "pseudo-p")[,6], "fdr") > 0.01
  g$lmc_vars_spdep <- droplevels(g$lmc_vars_spdep)
  if (tmap_ok) {
    if (tmap4) {
      tm_shape(g) + tm_polygons(fill="lmc_vars_spdep",
        fill.scale=tm_scale(values="brewer.set3", value.na="gray95",
          label.na="Insignificant"),
        fill.legend=tm_legend(position=tm_pos_in("left", "bottom"),
          frame=FALSE, item.r=0))
    } else {
      tm_shape(g) + tm_fill("lmc_vars_spdep", textNA="Insignificant",
        colorNA="gray95") + tm_borders() # rep. Fig. 11
    }
  } else {
    plot(g["lmc_vars_spdep"])
  }
}


## Not run:
library(reticulate)
use_python("/usr/bin/python", required = TRUE)
gp <- import("geopandas")
ps <- import("libpysal")
W <- listw2mat(listw)
w <- ps$weights$full2W(W, rownames(W))
w$transform <- "R"
esda <- import("esda")
lM <- esda$Moran_Local(x, w)
all.equal(unname(localmoran(x, listw, mlvar=FALSE)[,1]), c(lM$Is))
# confirm x and w the same
lC <- esda$Geary_Local(connectivity=w)$fit(scale(x))
# np$std missing ddof=1
n <- length(x)
D0 <- spdep:::geary.intern((x - mean(x)) / sqrt(var(x)*(n-1)/n), listw, n=n)
# lC components probably wrongly ordered https://github.com/pysal/esda/issues/192
o <- match(round(D0, 6), round(lC$localG, 6))
all.equal(c(lC$localG)[o], D0)
# simulation order not retained
lC$p_sim[o]
attr(C, "pseudo-p")[,6]

## End(Not run)
```

localG *G and Gstar local spatial statistics*

Description

The local spatial statistic G is calculated for each zone based on the spatial weights object used. The value returned is a Z-value, and may be used as a diagnostic tool. High positive values indicate the posibility of a local cluster of high values of the variable being analysed, very low relative values a similar cluster of low values. For inference, a Bonferroni-type test is suggested in the references, where tables of critical values may be found (see also details below).

Usage

```
localG(x, listw, zero.policy=attr(listw, "zero.policy"), spChk=NULL, GeoDa=FALSE,
 alternative = "two.sided", return_internals=TRUE)
localG_perm(x, listw, nsim=499, zero.policy=attr(listw, "zero.policy"), spChk=NULL,
 alternative = "two.sided", iseed=NULL, fix_i_in_Gstar_permutations=TRUE,
 no_repeat_in_row=FALSE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
zero.policy	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
GeoDa	default FALSE, if TRUE, drop x values for no-neighbour and self-neighbour only observations from all summations
nsim	default 499, number of conditonal permutation simulations
alternative	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
return_internals	default TRUE, unused
iseed	default NULL, used to set the seed; the output will only be reproducible if the count of CPU cores across which computation is distributed is the same
fix_i_in_Gstar_permutations	default TRUE (fix x at self in permutations for local G-star), set FALSE to use pre-1.2-8 behaviour
no_repeat_in_row	default FALSE, if TRUE, sample conditionally in each row without replacements to avoid duplicate values, https://github.com/r-spatial/spdep/issues/124

Details

If the neighbours member of listw has a "self.included" attribute set to TRUE, the Gstar variant, including the self-weight $w_{ii} > 0$, is calculated and returned. The returned vector will have a "gstari" attribute set to TRUE. Self-weights must be included by using the `include.self` function before converting the neighbour list to a spatial weights list with `nb2listw` as shown below in the example.

The critical values of the statistic under assumptions given in the references for the 95th percentile are for n=1: 1.645, n=50: 3.083, n=100: 3.289, n=1000: 3.886.

Value

A vector of G or Gstar standard deviate values, with attributes "gstari" set to TRUE or FALSE, "call" set to the function call, and class "localG". For conditional permutation, the returned value is the same as for `localG()`, and the simulated standard deviate is returned as column `"StdDev.Gi"` in `attr(., "internals")`.

Note

Conditional permutations added for comparative purposes; permutations are over the whole data vector omitting the observation itself, and from 1.2-8 fixing the observation itself as its own neighbour for local G-star.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Ord, J. K. and Getis, A. 1995 Local spatial autocorrelation statistics: distributional issues and an application. *Geographical Analysis*, 27, 286–306; Getis, A. and Ord, J. K. 1996 Local spatial statistics: an overview. In P. Longley and M. Batty (eds) *Spatial analysis: modelling in a GIS environment* (Cambridge: Geoinformation International), 261–277; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x

Examples

```
data(getisord, package="spData")
# spData 0.3.2 changes x, y, xyz object names to go_x, go_y, go_xyz to
# avoid putting these objects into the global environment via lazy loading
if (exists("go_xyz") && packageVersion("spData") >= "0.3.2") {
  xyz <- go_xyz
  x <- go_x
  y <- go_y
}
xycoords <- cbind(xyz$x, xyz$y)
nb30 <- dnearneigh(xycoords, 0, 30)
G30 <- localG(xyz$val, nb2listw(nb30, style="B"))
G30[length(xyz$val)-136]
set.seed(1)
```

```
G30_sim <- localG_perm(xyz$val, nb2listw(nb30, style="B"))
G30_sim[length(xyz$val)-136]
nb60 <- dnearneigh(xycoords, 0, 60)
G60 <- localG(xyz$val, nb2listw(nb60, style="B"))
G60[length(xyz$val)-136]
nb90 <- dnearneigh(xycoords, 0, 90)
G90 <- localG(xyz$val, nb2listw(nb90, style="B"))
G90[length(xyz$val)-136]
nb120 <- dnearneigh(xycoords, 0, 120)
G120 <- localG(xyz$val, nb2listw(nb120, style="B"))
G120[length(xyz$val)-136]
nb150 <- dnearneigh(xycoords, 0, 150)
G150 <- localG(xyz$val, nb2listw(nb150, style="B"))
G150[length(xyz$val)-136]
brks <- seq(-5,5,1)
cm.col <- cm.colors(length(brks)-1)
image(x, y, t(matrix(G30, nrow=16, ncol=16, byrow=TRUE)),
  breaks=brks, col=cm.col, asp=1)
text(xyz$x, xyz$y, round(G30, digits=1), cex=0.7)
polygon(c(195,225,225,195), c(195,195,225,225), lwd=2)
title(main=expression(paste("Values of the ", G[i], " statistic")))
G30s <- localG(xyz$val, nb2listw(include.self(nb30),
 style="B"))
cat("value according to Getis and Ord's eq. 14.2, p. 263 (1996)\n")
G30s[length(xyz$val)-136]
cat(paste("value given by Getis and Ord (1996), p. 267",
  "(division by n-1 rather than n \n in variance)\n"))
G30s[length(xyz$val)-136] *
  (sqrt(sum(scale(xyz$val, scale=FALSE)^2)/length(xyz$val)) /
  sqrt(var(xyz$val)))
image(x, y, t(matrix(G30s, nrow=16, ncol=16, byrow=TRUE)),
  breaks=brks, col=cm.col, asp=1)
text(xyz$x, xyz$y, round(G30s, digits=1), cex=0.7)
polygon(c(195,225,225,195), c(195,195,225,225), lwd=2)
title(main=expression(paste("Values of the ", G[i]^"*", " statistic")))
```

`localGS` *A local hotspot statistic for analysing multiscale datasets*

Description

The function implements the GS_i test statistic for local hotspots on specific pairwise evaluated distance bands, as proposed by Westerholt et al. (2015). Like the hotspot estimator G_i^*, the GS_i statistic is given as z-scores that can be evaluated accordingly. The idea of the method is to identify hotspots in datasets that comprise several, difficult-to-separate processes operating at different scales. This is often the case in complex user-generated datasets such as those from Twitter feeds. For example, a football match could be reflected in tweets from pubs, homes, and the stadium vicinity. These exemplified phenomena represent different processes that may be detected at different geometric scales. The GS_i method enables this identification by specifying a geometric scale band

and strictly calculating all statistical quantities such as mean and variance solely from respective relevant observations that interact on the range of the adjusted scale band. In addition, in each neighbourhood not only the relationships to the respective central unit, but all scale-relevant relationships are considered. In this way, hotspots can be detected on specific scale ranges independently of other scales. The statistic is given as:

$$GS_i = \frac{\sum_{j;k<j} w_{ij}w_{ik}\phi_{jk}a_{jk} - \frac{W_i}{\Phi}\sum_{j;k<j}\phi_{jk}a_{jk}}{\sqrt{\frac{W_i}{\Phi}\sum_{j;k<j}\phi_{jk}a_{jk}^2 + \frac{W_i(W_i-1)}{\Phi(\Phi-1)}\left(\Gamma^2 - \sum_{j;k<j}(\phi_{jk}a_{jk})^2\right) - \left(\frac{W_i}{\Phi}\sum_{j;k<j}\phi_{jk}a_{jk}\right)^2}}$$

with

$$a_{jk} = x_j + x_k, \quad W_i = \sum_{j;k<j} w_{ij}w_{ik}\phi_{jk}, \quad \Phi = \sum_{j;k<j}\phi_{jk}, \quad \text{and} \quad \Gamma = \sum_{j;k<j}\phi_{jk}a_{jk}.$$

Usage

```
localGS(x, listw, dmin, dmax, attr, longlat = NULL)
```

Arguments

x	a sf or sp object
listw	a listw object
dmin	a lower distance bound (greater than or equal)
dmax	an upper distance bound (less than or equal)
attr	the name of the attribute of interest
longlat	default NULL; TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometres; if x is a SpatialPoints object, the value is taken from the object itself, and overrides this argument if not NULL; distances are measured in map units if FALSE or NULL

Details

Only pairs of observations with a shared distance (in map units) on the interval [dmin, dmax] that are within a maximum radius of dmax around a corresponding output observation are considered. Thereby, also the mean values and variance terms estimated within the measure are adjusted to the scale range under consideration. For application examples of the method see Westerholt et al. (2015) (applied to tweets) and Sonea & Westerholt (2021) (applied in an access to banking scenario).

Value

A vector of GS_i values that are given as z-scores.

Author(s)

René Westerholt <rene.westerholt@tu-dortmund.de>

References

Westerholt, R., Resch, B. & Zipf, A. 2015. A local scale-sensitive indicator of spatial autocorrelation for assessing high-and low-value clusters in multiscale datasets. International Journal of Geographical Information Science, 29(5), 868–887, doi:10.1080/13658816.2014.1002499.

Sonea, A. and Westerholt, R. (2021): Geographic and temporal access to basic banking services in Wales. Applied Spatial Analysis and Policy, 14 (4), 879–905, doi:10.1007/s12061021093863.

See Also

localG

Examples

```
boston.tr <- sf::st_read(system.file("shapes/boston_tracts.gpkg", package="spData")[1])
boston.tr_utm <- st_transform(boston.tr, 32619) #26786

boston_listw1 <- nb2listwdist(dnearneigh(st_centroid(boston.tr_utm), 1, 2000),
    boston.tr_utm, type = "dpd", alpha = 2, zero.policy = TRUE, dmax = 9500)

boston_listw2 <- nb2listwdist(dnearneigh(st_centroid(boston.tr_utm), 5000, 9500),
    boston.tr_utm, type = "dpd", alpha = 2, zero.policy = TRUE, dmax = 9500)

boston_RM_gsi_1 <- localGS(boston.tr_utm, boston_listw1, 1, 2000, "RM", FALSE)
boston_RM_gsi_2 <- localGS(boston.tr_utm, boston_listw2, 2000, 9500, "RM", FALSE)
```

---

localmoran *Local Moran's I statistic*

Description

The local spatial statistic Moran's I is calculated for each zone based on the spatial weights object used. The values returned include a Z-value, and may be used as a diagnostic tool. The statistic is:

$$I_i = \frac{(x_i - \bar{x})}{\sum_{k=1}^{n}(x_k - \bar{x})^2/(n-1)} \sum_{j=1}^{n} w_{ij}(x_j - \bar{x})$$

, and its expectation and variance were given in Anselin (1995), but those from Sokal et al. (1998) are implemented here.

Usage

```
localmoran(x, listw, zero.policy=attr(listw, "zero.policy"), na.action=na.fail,
        conditional=TRUE, alternative = "two.sided", mlvar=TRUE,
        spChk=NULL, adjust.x=FALSE)
localmoran_perm(x, listw, nsim=499, zero.policy=attr(listw, "zero.policy"),
        na.action=na.fail, alternative = "two.sided", mlvar=TRUE,
        spChk=NULL, adjust.x=FALSE, sample_Ei=TRUE, iseed=NULL,
        no_repeat_in_row=FALSE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
zero.policy	default default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. If na.pass is used, zero is substituted for NA values in calculating the spatial lag. (Note that na.exclude will only work properly starting from R 1.9.0, na.omit and na.exclude assign the wrong classes in 1.8.*)
conditional	default TRUE: expectation and variance are calculated using the conditional randomization null (Sokal 1998 Eqs. A7 & A8). Elaboration of these changes available in Sauer et al. (2021). If FALSE: expectation and variance are calculated using the total randomization null (Sokal 1998 Eqs. A3 & A4).
alternative	a character string specifying the alternative hypothesis, must be one of greater, less or two.sided (default).
mlvar	default TRUE: values of local Moran's I are reported using the variance of the variable of interest (sum of squared deviances over n), but can be reported as the sample variance, dividing by (n-1) instead; both are used in other implementations.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.x	default FALSE, if TRUE, x values of observations with no neighbours are omitted in the mean of x
nsim	default 499, number of conditonal permutation simulations
sample_Ei	default TRUE; if conditional permutation, use the sample E_i values, or the analytical values, leaving only variances calculated by simulation.
iseed	default NULL, used to set the seed; the output will only be reproducible if the count of CPU cores across which computation is distributed is the same
no_repeat_in_row	default FALSE, if TRUE, sample conditionally in each row without replacements to avoid duplicate values, https://github.com/r-spatial/spdep/issues/124

Details

The values of local Moran's I are divided by the variance (or sample variance) of the variable of interest to accord with Table 1, p. 103, and formula (12), p. 99, in Anselin (1995), rather than his formula (7), p. 98. The variance of the local Moran statistic is taken from Sokal et al. (1998) p. 334, equations 4 & 5 or equations 7 & 8 located depending on user specification. By default, the implementation divides by n, not (n-1) in calculating the variance and higher moments. Conditional code contributed by Jeff Sauer and Levi Wolf.

Value

Ii	local moran statistic
E.Ii	expectation of local moran statistic; for localmoran_permthe permutation sample means
Var.Ii	variance of local moran statistic; for localmoran_permthe permutation sample standard deviations
Z.Ii	standard deviate of local moran statistic; for localmoran_perm based on permutation sample means and standard deviations
Pr()	p-value of local moran statistic using pnorm(); for localmoran_perm using standard deviatse based on permutation sample means and standard deviations
Pr() Sim	For localmoran_perm, rank() and punif() of observed statistic rank for [0, 1] p-values using alternative=
Pr(folded) Sim	the simulation folded [0, 0.5] range ranked p-value (based on https://github.com/pysal/esda/blob/4a63e0b5df1e754b17b5f1205b8cadcbecc5e061/esda/crand.py#L211-L213)
Skewness	For localmoran_perm, the output of e1071::skewness() for the permutation samples underlying the standard deviates
Kurtosis	For localmoran_perm, the output of e1071::kurtosis() for the permutation samples underlying the standard deviates

In addition, an attribute data frame "quadr" with mean and median quadrant columns, and a column splitting on the demeaned variable and lagged demeaned variable at zero.

Note

Conditional permutations added for comparative purposes; permutations are over the whole data vector omitting the observation itself. For p-value adjustment, use p.adjust() or p.adjustSP() on the output vector.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Anselin, L. 1995. Local indicators of spatial association, Geographical Analysis, 27, 93–115; Getis, A. and Ord, J. K. 1996 Local spatial statistics: an overview. In P. Longley and M. Batty (eds) *Spatial analysis: modelling in a GIS environment* (Cambridge: Geoinformation International), 261–277; Sokal, R. R, Oden, N. L. and Thomson, B. A. 1998. Local Spatial Autocorrelation in a Biological Model. Geographical Analysis, 30. 331–354; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x; Sauer, J., Oshan, T. M., Rey, S., & Wolf, L. J. 2021. The Importance of Null Hypotheses: Understanding Differences in Local Moran's under Heteroskedasticity. Geographical Analysis. doi:10.1111/gean.12304

Bivand, R. (2022), R Packages for Analyzing Spatial Data: A Comparative Case Study with Areal Data. Geographical Analysis, 54(3), 488-518. doi:10.1111/gean.12319

See Also

localG

Examples

```
data(afcon, package="spData")
oid <- order(afcon$id)
resI <- localmoran(afcon$totcon, nb2listw(paper.nb))
printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]),
 check.names=FALSE)
hist(resI[,5])
mean(resI[,1])
sum(resI[,1])/Szero(nb2listw(paper.nb))
moran.test(afcon$totcon, nb2listw(paper.nb))
# note equality for mean() only when the sum of weights equals
# the number of observations (thanks to Juergen Symanzik)
resI <- localmoran(afcon$totcon, nb2listw(paper.nb))
printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]),
 check.names=FALSE)
hist(p.adjust(resI[,5], method="bonferroni"))
totcon <-afcon$totcon
is.na(totcon) <- sample(1:length(totcon), 5)
totcon
resI.na <- localmoran(totcon, nb2listw(paper.nb), na.action=na.exclude,
 zero.policy=TRUE)
if (class(attr(resI.na, "na.action")) == "exclude") {
 print(data.frame(resI.na[oid,], row.names=afcon$name[oid]), digits=2)
} else print(resI.na, digits=2)
resG <- localG(afcon$totcon, nb2listw(include.self(paper.nb)))
print(data.frame(resG[oid], row.names=afcon$name[oid]), digits=2)
set.seed(1)
resI_p <- localmoran_perm(afcon$totcon, nb2listw(paper.nb))
printCoefmat(data.frame(resI_p[oid,], row.names=afcon$name[oid]),
 check.names=FALSE)
```

---

localmoran.exact *Exact local Moran's Ii tests*

Description

`localmoran.exact` provides exact local Moran's Ii tests under the null hypothesis, while `localmoran.exact.alt` provides exact local Moran's Ii tests under the alternative hypothesis. In this case, the model may be a fitted model derived from a model fitted by `spatialreg::errorsarlm`, with the covariance matrix can be passed through the `Omega=` argument.

Usage

```
localmoran.exact(model, select, nb, glist = NULL, style = "W",
 zero.policy = NULL, alternative = "two.sided", spChk = NULL,
```

```
 resfun = weighted.residuals, save.Vi = FALSE, useTP=FALSE, truncErr=1e-6,
 zeroTreat=0.1)
localmoran.exact.alt(model, select, nb, glist = NULL, style = "W",
 zero.policy = NULL, alternative = "two.sided", spChk = NULL,
 resfun = weighted.residuals, Omega = NULL, save.Vi = FALSE,
 save.M = FALSE, useTP=FALSE, truncErr=1e-6, zeroTreat=0.1)
## S3 method for class 'localmoranex'
print(x, ...)
## S3 method for class 'localmoranex'
as.data.frame(x, row.names=NULL, optional=FALSE, ...)
```

Arguments

model	an object of class lm returned by lm (assuming no global spatial autocorrelation), or an object of class sarlm returned by a spatial simultaneous autoregressive model fit (assuming global spatial autocorrelation represented by the model spatial coefficient); weights may be specified in the lm fit, but offsets should not be used
select	an integer vector of the id. numbers of zones to be tested; if missing, all zones
nb	a list of neighbours of class nb
glist	a list of general weights corresponding to neighbours
style	can take values W, B, C, and S
zero.policy	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
resfun	default: weighted.residuals; the function to be used to extract residuals from the lm object, may be residuals, weighted.residuals, rstandard, or rstudent
Omega	A SAR process matrix may be passed in to test an alternative hypothesis, for example Omega <- invIrW(listw, rho=0.1); Omega <- tcrossprod(Omega), chol() is taken internally
save.Vi	if TRUE, return the star-shaped weights lists for each zone tested
save.M	if TRUE, save a list of left and right M products
useTP	default FALSE, if TRUE, use truncation point in integration rather than upper=Inf, see Tiefelsdorf (2000), eq. 6.7, p.69
truncErr	when useTP=TRUE, pass truncation error to truncation point function
zeroTreat	when useTP=TRUE, pass zero adjustment to truncation point function
x	object to be printed
row.names	ignored argument to as.data.frame.localmoranex; row names assigned from localmoranex object
optional	ignored argument to as.data.frame.localmoranex; row names assigned from localmoranex object
...	arguments to be passed through

Value

A list with class `localmoranex` containing "select" lists, each with class `moranex` with the following components:

`statistic`	the value of the exact standard deviate of global Moran's I.
`p.value`	the p-value of the test.
`estimate`	the value of the observed local Moran's Ii.
`method`	a character string giving the method used.
`alternative`	a character string describing the alternative hypothesis.
`gamma`	eigenvalues (two extreme values for null, vector for alternative)
`oType`	usually set to "E", but set to "N" if the integration leads to an out of domain value for `qnorm`, when the Normal assumption is substituted. This only occurs when the output p-value would be very close to zero
`data.name`	a character string giving the name(s) of the data.
`df`	degrees of freedom
`i`	zone tested
`Vi`	zone tested

When the alternative is being tested, a list of left and right M products in attribute M.

Author(s)

Markus Reder and Roger Bivand

References

Bivand RS, Müller W, Reder M (2009) Power calculations for global and local Moran's I. Comput Stat Data Anal 53:2859–2872; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x

See Also

`lm.morantest.exact`, `localmoran.sad`

Examples

```
eire <- st_read(system.file("shapes/eire.gpkg", package="spData")[1])
row.names(eire) <- as.character(eire$names)
eire.nb <- poly2nb(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
localmoran.sad(e.lm, nb=eire.nb)
localmoran.exact(e.lm, nb=eire.nb)
localmoran.exact(e.lm, nb=eire.nb, useTP=TRUE)
run <- FALSE
if (requireNamespace("spatialreg", quietly=TRUE)) run <- TRUE
if (run) {
e.errorsar <- spatialreg::errorsarlm(OWNCONS ~ ROADACC, data=eire,
```

```
 listw=nb2listw(eire.nb))
lm.target <- lm(e.errorsar$tary ~ e.errorsar$tarX - 1)
localmoran.exact.alt(lm.target, nb=eire.nb)
}
if (run) {
Omega <- spatialreg::invIrW(nb2listw(eire.nb), rho=e.errorsar$lambda)
Omega1 <- tcrossprod(Omega)
localmoran.exact.alt(lm.target, nb=eire.nb, Omega=Omega1)
}
if (run) {
localmoran.exact.alt(lm.target, nb=eire.nb, Omega=Omega1, useTP=TRUE)
}
```

`localmoran.sad` *Saddlepoint approximation of local Moran's Ii tests*

Description

The function implements Tiefelsdorf's application of the Saddlepoint approximation to local Moran's Ii's reference distribution. If the model object is of class "lm", global independence is assumed; if of class "sarlm", global dependence is assumed to be represented by the spatial parameter of that model. Tests are reported separately for each zone selected, and may be summarised using `summary.localmoransad`. Values of local Moran's Ii agree with those from `localmoran()`, but in that function, the standard deviate - here the Saddlepoint approximation - is based on the randomisation assumption.

Usage

```
localmoran.sad(model, select, nb, glist=NULL, style="W",
 zero.policy=NULL, alternative="two.sided", spChk=NULL,
 resfun=weighted.residuals, save.Vi=FALSE,
 tol = .Machine$double.eps^0.5, maxiter = 1000, tol.bounds=0.0001,
 save.M=FALSE, Omega = NULL)
## S3 method for class 'localmoransad'
print(x, ...)
## S3 method for class 'localmoransad'
summary(object, ...)
## S3 method for class 'summary.localmoransad'
print(x, ...)
listw2star(listw, ireg, style, n, D, a, zero.policy=attr(listw, "zero.policy"))
```

Arguments

`model`	an object of class `lm` returned by `lm` (assuming no global spatial autocorrelation), or an object of class `sarlm` returned by a spatial simultaneous autoregressive model fit (assuming global spatial autocorrelation represented by the model spatial coefficient); weights may be specified in the `lm` fit, but offsets should not be used

select	an integer vector of the id. numbers of zones to be tested; if missing, all zones
nb	a list of neighbours of class nb
glist	a list of general weights corresponding to neighbours
style	can take values W, B, C, and S
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
resfun	default: weighted.residuals; the function to be used to extract residuals from the lm object, may be residuals, weighted.residuals, rstandard, or rstudent
save.Vi	if TRUE, return the star-shaped weights lists for each zone tested
tol	the desired accuracy (convergence tolerance) for uniroot
maxiter	the maximum number of iterations for uniroot
tol.bounds	offset from bounds for uniroot
save.M	if TRUE, save a list of left and right M products in a list for the conditional tests, or a list of the regression model matrix components
Omega	A SAR process matrix may be passed in to test an alternative hypothesis, for example Omega <- invIrW(listw, rho=0.1); Omega <- tcrossprod(Omega), chol() is taken internally
x	object to be printed
object	object to be summarised
...	arguments to be passed through
listw	a listw object created for example by nb2listw
ireg	a zone number
n	internal value depending on listw and style
D	internal value depending on listw and style
a	internal value depending on listw and style

Details

The function implements the analytical eigenvalue calculation together with trace shortcuts given or suggested in Tiefelsdorf (2002), partly following remarks by J. Keith Ord, and uses the Saddlepoint analytical solution from Tiefelsdorf's SPSS code.

If a histogram of the probability values of the saddlepoint estimate for the assumption of global independence is not approximately flat, the assumption is probably unjustified, and re-estimation with global dependence is recommended.

No n by n matrices are needed at any point for the test assuming no global dependence, the star-shaped weights matrices being handled as listw lists. When the test is made on residuals from a spatial regression, taking a global process into account. n by n matrices are necessary, and memory constraints may be reached for large lattices.

Value

A list with class `localmoransad` containing "select" lists, each with class `moransad` with the following components:

`statistic`	the value of the saddlepoint approximation of the standard deviate of local Moran's Ii.
`p.value`	the p-value of the test.
`estimate`	the value of the observed local Moran's Ii.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string giving the method used.
`data.name`	a character string giving the name(s) of the data.
`internal1`	Saddlepoint omega, r and u
`df`	degrees of freedom
`tau`	maximum and minimum analytical eigenvalues
`i`	zone tested

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Tiefelsdorf, M. 2002 The Saddlepoint approximation of Moran's I and local Moran's Ii reference distributions and their numerical evaluation. Geographical Analysis, 34, pp. 187–206.

See Also

`localmoran`, `lm.morantest`, `lm.morantest.sad`, `errorsarlm`

Examples

```
eire <- st_read(system.file("shapes/eire.gpkg", package="spData")[1])
row.names(eire) <- as.character(eire$names)
eire.nb <- poly2nb(eire)
lw <- nb2listw(eire.nb)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
e.locmor <- summary(localmoran.sad(e.lm, nb=eire.nb))
e.locmor
mean(e.locmor[,1])
sum(e.locmor[,1])/Szero(lw)
lm.morantest(e.lm, lw)
# note equality for mean() only when the sum of weights equals
# the number of observations (thanks to Juergen Symanzik)
hist(e.locmor[,"Pr. (Sad)"])
e.wlm <- lm(OWNCONS ~ ROADACC, data=eire, weights=RETSALE)
e.locmorw1 <- summary(localmoran.sad(e.wlm, nb=eire.nb, resfun=weighted.residuals))
e.locmorw1
e.locmorw2 <- summary(localmoran.sad(e.wlm, nb=eire.nb, resfun=rstudent))
```

```
e.locmorw2
run <- FALSE
if (requireNamespace("spatialreg", quietly=TRUE)) run <- TRUE
if (run) {
e.errorsar <- spatialreg::errorsarlm(OWNCONS ~ ROADACC, data=eire,
  listw=lw)
if (packageVersion("spatialreg") < "1.1.7")
  spatialreg::print.sarlm(e.errorsar)
else
  print(e.errorsar)
}
if (run) {
lm.target <- lm(e.errorsar$tary ~ e.errorsar$tarX - 1)
Omega <- tcrossprod(spatialreg::invIrW(lw, rho=e.errorsar$lambda))
e.clocmor <- summary(localmoran.sad(lm.target, nb=eire.nb, Omega=Omega))
e.clocmor
}
if (run) {
hist(e.clocmor[,"Pr. (Sad)"])
}
```

`localmoran_bv` *Compute the Local Bivariate Moran's I Statistic*

Description

Given two continuous numeric variables, calculate the bivariate Local Moran's I.

Usage

```
localmoran_bv(x, y, listw, nsim = 199, scale = TRUE, alternative="two.sided",
 iseed=1L, no_repeat_in_row=FALSE, zero.policy=attr(listw, "zero.policy"))
```

Arguments

`x`	a numeric vector of same length as `y`.
`y`	a numeric vector of same length as `x`.
`listw`	a listw object for example as created by `nb2listw()`.
`nsim`	the number of simulations to run.
`scale`	default `TRUE`.
`alternative`	a character string specifying the alternative hypothesis, must be one of "greater" (default), "two.sided", or "less".
`iseed`	default NULL, used to set the seed; the output will only be reproducible if the count of CPU cores across which computation is distributed is the same.
`no_repeat_in_row`	default `FALSE`, if `TRUE`, sample conditionally in each row without replacements to avoid duplicate values, https://github.com/r-spatial/spdep/issues/124

`zero.policy` default default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

Details

The Bivariate Local Moran, like its global counterpart, evaluates the value of x at observation i with its spatial neighbors' value of y. The value of I_i^B is xi * Wyi. Or, in simpler words, the local bivariate Moran is the result of multiplying x by the spatial lag of y. Formally it is defined as

$I_i^B = cx_i \Sigma_j w_{ij} y_j$

Value

a `data.frame` containing two columns `Ib` and `p_sim` containing the local bivariate Moran's I and simulated p-values respectively.

Author(s)

Josiah Parry <josiah.parry@gmail.com>

References

Anselin, Luc, Ibnu Syabri, and Oleg Smirnov. 2002. "Visualizing Multivariate Spatial Correlation with Dynamically Linked Windows." In New Tools for Spatial Data Analysis: Proceedings of the Specialist Meeting, edited by Luc Anselin and Sergio Rey. University of California, Santa Barbara: Center for Spatially Integrated Social Science (CSISS).

Examples

```
# load columbus datay
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData"))
nb <- poly2nb(columbus)
listw <- nb2listw(nb)
set.seed(1)
(res <- localmoran_bv(columbus$CRIME, columbus$INC, listw, nsim = 499))
columbus$hs <- hotspot(res, Prname="Pr(folded) Sim", cutoff=0.05,
 quadrant.type="pysal", p.adjust="none")

if (require("tmap", quietly=TRUE)) {
tmap4 <- packageVersion("tmap") >= "3.99"
if (tmap4) {
  tm_shape(columbus) + tm_polygons(fill="hs",
    fill.scale=tm_scale(values="brewer.set3"),
    fill.legend=tm_legend(position=tm_pos_in("left", "top"),
      frame=FALSE, item.r=0), lwd=0.01)
} else {
  tm_shape(columbus) + tm_fill("hs")
}
}

moran.plot(x=columbus$CRIME, y=columbus$INC, listw=listw)
```

`local_joincount_bv` *Calculate the local bivariate join count*

Description

The bivariate join count (BJC) evaluates event occurrences in predefined regions and tests if the co-occurrence of events deviates from complete spatial randomness.

Usage

```
local_joincount_bv(
  x,
  z,
  listw,
  nsim = 199,
  alternative = "two.sided"
)
```

Arguments

`x`	a binary variable either numeric or logical
`z`	a binary variable either numeric or logical with the same length as `x`
`listw`	a listw object containing binary weights created, for example, with `nb2listw(nb, style = "B")`
`nsim`	the number of conditional permutation simulations
`alternative`	default `"greater"`. One of `"less"` or `"greater"`.

Details

There are two cases that are evaluated in the bivariate join count. The first being in-situ colocation (CLC) where xi = 1 and zi = 1. The second is the general form of the bivariate join count (BJC) that is used when there is no in-situ colocation.

The BJC case "is useful when x and z cannot occur in the same location, such as when x and z correspond to two different values of a single categorical variable" or "when x and z can co-locate, but do not" (Anselin and Li, 2019). Whereas the CLC case is useful in evaluating simultaneous occurrences of events.

The local bivariate join count statistic requires a binary weights list which can be generated with `nb2listw(nb, style = "B")`.

P-values are only reported for those regions that match the CLC or BJC criteria. Others will not have an associated p-value.

P-values are estimated using a conditional permutation approach. This creates a reference distribution from which the observed statistic is compared.

Value

a `data.frame` with two columns `join_count` and `p_sim` and number of rows equal to the length of arguments `x`.

Author(s)

Josiah Parry <josiah.parry@gmail.com>

References

Anselin, L., & Li, X. (2019). Operational Local Join Count Statistics for Cluster Detection. Journal of geographical systems, 21(2), 189–210. doi:10.1007/s1010901900299x

Examples

```
data("oldcol")
listw <- nb2listw(COL.nb, style = "B")
# Colocation case
x <- COL.OLD[["CP"]]
z <- COL.OLD[["EW"]]
set.seed(1)
res <- local_joincount_bv(x, z, listw)
na.omit(res)
# no colocation case
z <- 1 - x
set.seed(1)
res <- local_joincount_bv(x, z, listw)
na.omit(res)
```

---

local_joincount_uni *Calculate the local univariate join count*

Description

The univariate local join count statistic is used to identify clusters of rarely occurring binary variables. The binary variable of interest should occur less than half of the time.

Usage

```
local_joincount_uni(
  fx,
  chosen,
  listw,
  alternative = "two.sided",
  nsim = 199,
  iseed = NULL,
  no_repeat_in_row=FALSE
)
```

Arguments

fx	a binary variable either numeric or logical
chosen	a scalar character containing the level of fx that should be considered the observed value (1).
listw	a listw object containing binary weights created, for example, with nbwlistw(nb, style = "B")
alternative	default "greater". One of "less" or "greater".
nsim	the number of conditional permutation simulations
iseed	default NULL, used to set the seed; the output will only be reproducible if the count of CPU cores across which computation is distributed is the same
no_repeat_in_row	default FALSE, if TRUE, sample conditionally in each row without replacements to avoid duplicate values, https://github.com/r-spatial/spdep/issues/124

Details

The local join count statistic requires a binary weights list which can be generated with nb2listw(nb, style = "B"). Additionally, ensure that the binary variable of interest is rarely occurring in no more than half of observations.

P-values are estimated using a conditional permutation approach. This creates a reference distribution from which the observed statistic is compared. For more see Geoda Glossary.

Value

a data.frame with two columns BB and Pr() and number of rows equal to the length of x.

Author(s)

Josiah Parry <josiah.parry@gmail.com>

References

Anselin, L., & Li, X. (2019). Operational Local Join Count Statistics for Cluster Detection. Journal of geographical systems, 21(2), 189–210. doi:10.1007/s1010901900299x

Examples

```
data(oldcol)
fx <- as.factor(ifelse(COL.OLD$CRIME < 35, "low-crime", "high-crime"))
listw <- nb2listw(COL.nb, style = "B")
set.seed(1)
(res <- local_joincount_uni(fx, chosen = "high-crime", listw))
```

LOSH *Local spatial heteroscedasticity*

Description

Local spatial heteroscedasticity is calculated for each location based on the spatial weights object used. The statistic is:

$$H_i = \frac{\sum_j^n w_{ij} \cdot |e_j|^a}{h_1 \cdot \sum_j^n w_{ij}}$$

with

$$e_j = x_j - \bar{x}_j$$

and

$$\bar{x}_j = \frac{\sum_k^n w_{jk} \cdot x_k}{\sum_k^n w_{jk}}$$

Its expectation and variance are given in Ord & Getis (2012). The exponent *a* allows for investigating different types of mean dispersal.

Usage

```
LOSH(x, listw, a=2, var_hi=TRUE, zero.policy=attr(listw, "zero.policy"),
na.action=na.fail, spChk=NULL)
```

Arguments

x	a numeric vector of the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
a	the exponent applied to the local residuals; the default value of 2 leads to a measure of heterogeneity in the spatial variance
var_hi	default TRUE, the moments and the test statistics are calculated for each location; if FALSE, only the plain LOSH measures, $\bar{x}_i$ and e_i are calculated
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. If na.pass is used, zero is substituted for NA values in calculating the spatial lag. (Note that na.exclude will only work properly starting from R 1.9.0, na.omit and na.exclude assign the wrong classes in 1.8.*)
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()

Details

In addition to the LOSH measure, the values returned include local spatially weighted mean values $\bar{x}_i$ and local residuals e_i estimated about these means. These values facilitate the interpretation of LOSH values. Further, if specified through `var_hi`, the statistical moments and the test statistics as proposed by Ord & Getis (2012) are also calculated and returned.

Value

`Hi`	LOSH statistic
`E.Hi`	(optional) expectation of LOSH
`Var.Hi`	(optional) variance of LOSH
`Z.Hi`	(optional) the approximately Chi-square distributed test statistics
`x_bar_i`	local spatially weighted mean values
`ei`	residuals about local spatially weighted mean values

Author(s)

René Westerholt <rene.westerholt@tu-dortmund.de>

References

Ord, J. K., & Getis, A. 2012. Local spatial heteroscedasticity (LOSH), The Annals of Regional Science, 48 (2), 529–539.

See Also

`LOSH.cs`, `LOSH.mc`

Examples

```
data(boston, package="spData")
resLOSH <- LOSH(boston.c$NOX, nb2listw(boston.soi))
hist(resLOSH[,"Hi"])
mean(resLOSH[,"Hi"])
```

LOSH.cs *Chi-square based test for local spatial heteroscedasticity*

Description

The function implements the chi-square based test statistic for local spatial heteroscedasticity (LOSH) as proposed by Ord & Getis (2012).

Usage

```
LOSH.cs(x, listw, zero.policy = attr(listw, "zero.policy"), na.action = na.fail,
                 p.adjust.method = "none", spChk = NULL)
```

Arguments

x	a numeric vector of the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. If na.pass is used, zero is substituted for NA values in calculating the spatial lag. (Note that na.exclude will only work properly starting from R 1.9.0, na.omit and na.exclude assign the wrong classes in 1.8.*)
p.adjust.method	a character string specifying the probability value adjustment for multiple tests, default "none"; see p.adjustSP. Note that the number of multiple tests for each region is only taken as the number of neighbours + 1 for each region, rather than the total number of regions.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()

Details

The test uses $a = 2$ (see LOSH) because chi-square based inference is not applicable with other exponents. The function makes use of LOSH in its calculations.

Value

Hi	LOSH statistic
E.Hi	expectation of LOSH
Var.Hi	variance of LOSH
Z.Hi	the approximately chi-square distributed test statistics
x_bar_i	local spatially weighted mean values
ei	residuals about local spatially weighted mean values
Pr()	p-values for Hi obtained from a non-central Chi-square distribution with $2/Var.Hi$ degrees of freedom

Author(s)

René Westerholt <rene.westerholt@tu-dortmund.de>

References

Ord, J. K., & Getis, A. 2012. Local spatial heteroscedasticity (LOSH), The Annals of Regional Science, 48 (2), 529–539.

See Also

LOSH, LOSH.mc

Examples

```
data(boston, package="spData")
resLOSH <- LOSH.cs(boston.c$NOX, nb2listw(boston.soi))
hist(resLOSH[,"Hi"])
mean(resLOSH[,"Hi"])
```

LOSH.mc *Bootstrapping-based test for local spatial heteroscedasticity*

Description

The function draws inferences about local spatial heteroscedasticity (LOSH) by means of the randomisation-based Monte-Carlo bootstrap proposed by Xu et al. (2014).

Usage

```
LOSH.mc(x, listw, a = 2, nsim = 99, zero.policy = attr(listw, "zero.policy"),
 na.action = na.fail, spChk = NULL, adjust.n = TRUE, p.adjust.method = "none")
```

Arguments

x	a numeric vector of the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
a	the exponent applied to the local residuals; the default value of 2 leads to a measure of heterogeneity in the spatial variance
nsim	the number of randomisations used in the bootstrap
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. If na.pass is used, zero is substituted for NA values in calculating the spatial lag. (Note that na.exclude will only work properly starting from R 1.9.0, na.omit and na.exclude assign the wrong classes in 1.8.*)

spChk — should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`

adjust.n — default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

p.adjust.method — a character string specifying the probability value adjustment for multiple tests, default "none"; see `p.adjustSP`. Note that the number of multiple tests for each region is only taken as the number of neighbours + 1 for each region, rather than the total number of regions.

Details

The test calculates LOSH (see `LOSH`) and estimates pseudo p-values from a conditional bootstrap. Thereby, the i-th value in each location is held fixed, whereas all other values are permuted `nsim` times over all other spatial units.

Value

`Hi`	LOSH statistic
`E.Hi`	expectation of LOSH
`Var.Hi`	variance of LOSH
`Z.Hi`	the approximately chi-square distributed test statistics
`x_bar_i`	local spatially weighted mean values
`ei`	residuals about local spatially weighted mean values
`Pr()`	p-values for `Hi` obtained from a conditional bootstrap distribution

Author(s)

René Westerholt <rene.westerholt@tu-dortmund.de>

References

Ord, J. K., & Getis, A. 2012. Local spatial heteroscedasticity (LOSH), The Annals of Regional Science, 48 (2), 529–539; Xu, M., Mei, C. L., & Yan, N. 2014. A note on the null distribution of the local spatial heteroscedasticity (LOSH) statistic. The Annals of Regional Science, 52 (3), 697–710.

See Also

`LOSH`, `LOSH.mc`

Examples

```
data(columbus, package="spData")
resLOSH_mc <- LOSH.mc(columbus$CRIME, nb2listw(col.gal.nb), 2, 100)
summary(resLOSH_mc)
resLOSH_cs <- LOSH.cs(columbus$CRIME, nb2listw(col.gal.nb))
summary(resLOSH_cs)
plot(resLOSH_mc[,"Pr()"], resLOSH_cs[,"Pr()"])
```

`mat2listw` *Convert a square spatial weights matrix to a weights list object*

Description

The function converts a square spatial weights matrix, optionally a sparse matrix to a weights list object, optionally adding region IDs from the row names of the matrix, as a sequence of numbers 1:nrow(x), or as given as an argument. The style can be imposed by rebuilting the weights list object internally.

Usage

```
mat2listw(x, row.names = NULL, style=NULL, zero.policy = NULL)
```

Arguments

`x`	A square non-negative matrix with no NAs representing spatial weights; may be a matrix of class "sparseMatrix"
`row.names`	row names to use for region IDs
`style`	default NULL, missing, set to "M" and warning given; if not "M", passed to `nb2listw` to re-build the object
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors

Value

A `listw` object with the following members:

`style`	"M", meaning matrix style, underlying style unknown, or assigned style argument in rebuilt object
`neighbours`	the derived neighbours list
`weights`	the weights for the neighbours derived from the matrix

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`nb2listw`, `nb2mat`

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col005 <- dnearneigh(st_coordinates(st_centroid(st_geometry(columbus),
 of_largest_polygon=TRUE)), 0, 0.5, as.character(columbus$NEIGNO))
summary(col005)
col005.w.mat <- nb2mat(col005, style="W", zero.policy=TRUE)
try(col005.w.b <- mat2listw(col005.w.mat, style="W"))
col005.w.b <- mat2listw(col005.w.mat, style="W", zero.policy=TRUE)
summary(col005.w.b$neighbours)
diffnb(col005, col005.w.b$neighbours)
col005.w.mat.3T <- kronecker(diag(3), col005.w.mat)
col005.w.b.3T <- mat2listw(col005.w.mat.3T, style="W", zero.policy=TRUE)
summary(col005.w.b.3T$neighbours)
run <- FALSE
if (require("spatialreg", quiet=TRUE)) run <- TRUE
if (run) {
W <- as(nb2listw(col005, style="W", zero.policy=TRUE), "CsparseMatrix")
try(col005.spM <- mat2listw(W))
col005.spM <- mat2listw(W, style="W", zero.policy=TRUE)
summary(col005.spM$neighbours)
}
if (run) {
diffnb(col005, col005.spM$neighbours)
}
if (run && require("Matrix", quiet=TRUE)) {
IW <- kronecker(Diagonal(3), W)
col005.spM.3T <- mat2listw(as(IW, "CsparseMatrix"), style="W", zero.policy=TRUE)
summary(col005.spM.3T$neighbours)
}
```

`moran` *Compute Moran's I*

Description

A simple function to compute Moran's I, called by `moran.test` and `moran.mc`;

$$I = \frac{n}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Usage

```
moran(x, listw, n, S0, zero.policy=attr(listw, "zero.policy"), NAOK=FALSE)
```

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`

n	number of zones
S0	global sum of weights
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
NAOK	if 'TRUE' then any 'NA' or 'NaN' or 'Inf' values in x are passed on to the foreign function. If 'FALSE', the presence of 'NA' or 'NaN' or 'Inf' values is regarded as an error.

Value

a list of

I	Moran's I
K	sample kurtosis of x

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 17.

See Also

moran.test, moran.mc

Examples

```
data(oldcol)
col.W <- nb2listw(COL.nb, style="W")
crime <- COL.OLD$CRIME
str(moran(crime, col.W, length(COL.nb), Szero(col.W)))
is.na(crime) <- sample(1:length(crime), 10)
str(moran(crime, col.W, length(COL.nb), Szero(col.W), NAOK=TRUE))
```

---

moran.mc *Permutation test for Moran's I statistic*

Description

A permutation test for Moran's I statistic calculated by using nsim random permutations of x for the given spatial weighting scheme, to establish the rank of the observed statistic in relation to the nsim simulated values.

Usage

```
moran.mc(x, listw, nsim, zero.policy=attr(listw, "zero.policy"),
 alternative="greater", na.action=na.fail, spChk=NULL, return_boot=FALSE,
 adjust.n=TRUE)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
nsim	number of permutations
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "two.sided", or "less".
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. na.pass is not permitted because it is meaningless in a permutation test.
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
return_boot	return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

Value

A list with class htest and mc.sim containing the following components:

statistic	the value of the observed Moran's I.
parameter	the rank of the observed Moran's I.
p.value	the pseudo p-value of the test.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data, and the number of simulations.
res	nsim simulated values of statistic, final value is observed statistic

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 63-5.

See Also

moran, moran.test

Examples

```
data(oldcol)
colw <- nb2listw(COL.nb, style="W")
nsim <- 99
set.seed(1234)
sim1 <- moran.mc(COL.OLD$CRIME, listw=colw, nsim=nsim)
sim1
mean(sim1$res[1:nsim])
var(sim1$res[1:nsim])
summary(sim1$res[1:nsim])
colold.lags <- nblag(COL.nb, 3)
set.seed(1234)
sim2 <- moran.mc(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"), nsim=nsim)
summary(sim2$res[1:nsim])
sim3 <- moran.mc(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"), nsim=nsim)
summary(sim3$res[1:nsim])
crime <- COL.OLD$CRIME
is.na(crime) <- sample(1:length(crime), 10)
try(moran.mc(crime, nb2listw(COL.nb, style="W"), nsim=99,
 na.action=na.fail))
moran.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 na.action=na.omit)
moran.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 return_boot=TRUE, na.action=na.omit)
moran.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 na.action=na.exclude)
moran.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, zero.policy=TRUE,
 return_boot=TRUE, na.action=na.exclude)
try(moran.mc(crime, nb2listw(COL.nb, style="W"), nsim=99, na.action=na.pass))
```

---

moran.plot *Moran scatterplot*

Description

A plot of spatial data against its spatially lagged values, augmented by reporting the summary of influence measures for the linear relationship between the data and the lag. If zero policy is TRUE, such observations are also marked if they occur.

Usage

```
moran.plot(x, listw, y=NULL, zero.policy=attr(listw, "zero.policy"), spChk=NULL,
 labels=NULL, xlab=NULL, ylab=NULL, quiet=NULL, plot=TRUE, return_df=TRUE, ...)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
y	an optional numeric vector the same length as the neighbours list in listw for a bi-variate plot
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
labels	character labels for points with high influence measures, if set to FALSE, no labels are plotted for points with large influence
xlab	label for x axis
ylab	label for x axis
quiet	default NULL, use !verbose global option value; if TRUE, output of summary of influence object suppressed
plot	default TRUE, if false, plotting is suppressed
return_df	default TRUE, invisibly return a data.frame object; if FALSE invisibly return an influence measures object
...	further graphical parameters as in par(..)

Value

The function returns a data.frame object with coordinates and influence measures if return_df is TRUE, or an influence object from influence.measures.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Anselin, L. 1996. The Moran scatterplot as an ESDA tool to assess local instability in spatial association. pp. 111–125 in M. M. Fischer, H. J. Scholten and D. Unwin (eds) Spatial analytical perspectives on GIS, London, Taylor and Francis; Anselin, L. 1995. Local indicators of spatial association, Geographical Analysis, 27, 93–115

See Also

localmoran, influence.measures

Examples

```
data(afcon, package="spData")
mp <- moran.plot(afcon$totcon, nb2listw(paper.nb),
 labels=as.character(afcon$name), pch=19)
moran.plot(as.vector(scale(afcon$totcon)), nb2listw(paper.nb),
 labels=as.character(afcon$name), xlim=c(-2, 4), ylim=c(-2,4), pch=19)
if (require(ggplot2, quietly=TRUE)) {
  xname <- attr(mp, "xname")
  ggplot(mp, aes(x=x, y=wx)) + geom_point(shape=1) +
    geom_smooth(formula=y ~ x, method="lm") +
    geom_hline(yintercept=mean(mp$wx), lty=2) +
    geom_vline(xintercept=mean(mp$x), lty=2) + theme_minimal() +
    geom_point(data=mp[mp$is_inf,], aes(x=x, y=wx), shape=9) +
    geom_text(data=mp[mp$is_inf,], aes(x=x, y=wx, label=labels, vjust=1.5)) +
    xlab(xname) + ylab(paste0("Spatially lagged ", xname))
}
columbus <- st_read(system.file("shapes/columbus.shp", package="spData"))
nb <- poly2nb(columbus)
listw <- nb2listw(nb)
moran.plot(x=columbus$CRIME, y=columbus$INC, listw=listw)
moran.plot(x=columbus$INC, y=columbus$CRIME, listw=listw)
```

---

moran.test *Moran's I test for spatial autocorrelation*

Description

Moran's test for spatial autocorrelation using a spatial weights matrix in weights list form. The assumptions underlying the test are sensitive to the form of the graph of neighbour relationships and other factors, and results may be checked against those of `moran.mc` permutations.

Usage

```
moran.test(x, listw, randomisation=TRUE, zero.policy=attr(listw, "zero.policy"),
 alternative="greater", rank = FALSE, na.action=na.fail, spChk=NULL,
 adjust.n=TRUE, drop.EI2=FALSE)
```

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`
`randomisation`	variance of I calculated under the assumption of randomisation, if FALSE normality
`zero.policy`	default `attr(listw, "zero.policy")` as set when `listw` was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

alternative	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
rank	logical value - default FALSE for continuous variables, if TRUE, uses the adaptation of Moran's I for ranks suggested by Cliff and Ord (1981, p. 46)
na.action	a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. If na.pass is used, zero is substituted for NA values in calculating the spatial lag
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
drop.EI2	default FALSE, if TRUE, emulate CrimeStat <= 4.02

Value

A list with class htest containing the following components:

statistic	the value of the standard deviate of Moran's I.
p.value	the p-value of the test.
estimate	the value of the observed Moran's I, its expectation and variance under the method assumption.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the assumption used for calculating the standard deviate.
data.name	a character string giving the name(s) of the data.

Note

Var(I) is taken from Cliff and Ord (1969, p. 28), and Goodchild's CATMOG 47 (1986), see also Upton & Fingleton (1985) p. 171; it agrees with SpaceStat, see Tutorial workbook Chapter 22; VI is the second crude moment minus the square of the first crude moment. The derivation of the test (Cliff and Ord, 1981, p. 18) assumes that the weights matrix is symmetric. For inherently non-symmetric matrices, such as k-nearest neighbour matrices, listw2U() can be used to make the matrix symmetric.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 21; Bivand RS, Wong DWS 2018 Comparing implementations of global and local indicators of spatial association. TEST, 27(3), 716–748 doi:10.1007/s117490180599x

See Also

moran, moran.mc, listw2U

Examples

```
data(oldcol)
coords.OLD <- cbind(COL.OLD$X, COL.OLD$Y)
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="B"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="C"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="S"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
colold.lags <- nblag(COL.nb, 3)
moran.test(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"))
moran.test(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"))
print(is.symmetric.nb(COL.nb))
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
moran.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"))
moran.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"),
 randomisation=FALSE)
cat("Note: non-symmetric weights matrix, use listw2U()")
moran.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")))
moran.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")), randomisation=FALSE)
ranks <- rank(COL.OLD$CRIME)
names(ranks) <- rownames(COL.OLD)
moran.test(ranks, nb2listw(COL.nb, style="W"), rank=TRUE)
crime <- COL.OLD$CRIME
is.na(crime) <- sample(1:length(crime), 10)
res <- try(moran.test(crime, nb2listw(COL.nb, style="W"),
 na.action=na.fail))
res
moran.test(crime, nb2listw(COL.nb, style="W"), zero.policy=TRUE,
 na.action=na.omit)
moran.test(crime, nb2listw(COL.nb, style="W"), zero.policy=TRUE,
 na.action=na.exclude)
moran.test(crime, nb2listw(COL.nb, style="W"), na.action=na.pass)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col_geoms <- st_geometry(columbus)
col_geoms[1] <- st_buffer(col_geoms[1], dist=-0.05)
st_geometry(columbus) <- col_geoms
(nb1 <- poly2nb(columbus))
try(lw <- nb2listw(nb1, style="W"))
(lw <- nb2listw(nb1, style="W", zero.policy=TRUE))
moran.test(COL.OLD$CRIME, lw)
```

`moran_bv` *Compute the Global Bivariate Moran's I*

Description

Given two continuous numeric variables, calculate the bivariate Moran's I. See details for more.

Usage

```
moran_bv(x, y, listw, nsim = 499, scale = TRUE)
```

Arguments

`x`	a numeric vector of same length as `y`.
`y`	a numeric vector of same length as `x`.
`listw`	a listw object for example as created by `nb2listw()`.
`nsim`	the number of simulations to run.
`scale`	default `TRUE`.

Details

The Global Bivariate Moran is defined as

$I_B = \frac{\Sigma_i(\Sigma_j w_{ij} y_j \times x_i)}{\Sigma_i x_i^2}$

It is important to note that this is a measure of autocorrelation of X with the spatial lag of Y. As such, the resultant measure may overestimate the amount of spatial autocorrelation which may be a product of the inherent correlation of X and Y. The output object is of class `"boot"`, so that plots and confidence intervals are available using appropriate methods.

Value

An object of class `"boot"`, with the observed statistic in component `t0`.

Author(s)

Josiah Parry `<josiah.parry@gmail.com>`

References

Wartenberg, D. (1985), Multivariate Spatial Correlation: A Method for Exploratory Geographical Analysis. Geographical Analysis, 17: 263-283. doi:10.1111/j.15384632.1985.tb00849.x

Examples

```
data(boston, package = "spData")
x <- boston.c$CRIM
y <- boston.c$NOX
listw <- nb2listw(boston.soi)
set.seed(1)
res_xy <- moran_bv(x, y, listw, nsim=499)
res_xy$t0
boot::boot.ci(res_xy, conf=c(0.99, 0.95, 0.9), type="basic")
plot(res_xy)
set.seed(1)
lee_xy <- lee.mc(x, y, listw, nsim=499, return_boot=TRUE)
lee_xy$t0
boot::boot.ci(lee_xy, conf=c(0.99, 0.95, 0.9), type="basic")
plot(lee_xy)
set.seed(1)
res_yx <- moran_bv(y, x, listw, nsim=499)
res_yx$t0
boot::boot.ci(res_yx, conf=c(0.99, 0.95, 0.9), type="basic")
plot(res_yx)
set.seed(1)
lee_yx <- lee.mc(y, x, listw, nsim=499, return_boot=TRUE)
lee_yx$t0
boot::boot.ci(lee_yx, conf=c(0.99, 0.95, 0.9), type="basic")
plot(lee_yx)
```

`mstree` *Find the minimal spanning tree*

Description

The minimal spanning tree is a connected graph with n nodes and n-1 edges. This is a smaller class of possible partitions of a graph by pruning edges with high dissimilarity. If one edge is removed, the graph is partioned in two unconnected subgraphs. This function implements the algorithm due to Prim (1987).

Usage

```
mstree(nbw, ini = NULL)
```

Arguments

`nbw`	An object of `listw` class returned by `nb2listw` function. See this help for details.
`ini`	The initial node in the minimal spanning tree.

Details

The minimum spanning tree algorithm.

Input a connected graph.

Begin a empty set of nodes.

Add an arbitrary note in this set.

While are nodes not in the set, find a minimum cost edge connecting a node in the set and a node out of the set and add this node in the set.

The set of edges is a minimum spanning tree.

Value

A matrix with n-1 rows and tree columns. Each row is two nodes and the cost, i. e. the edge and it cost.

Author(s)

Renato M. Assuncao and Elias T. Krainski

References

R. C. Prim (1957) Shortest connection networks and some generalisations. In: Bell System Technical Journal, 36, pp. 1389-1401

Examples

```
### loading data
(GDAL37 <- as.numeric_version(unname(sf_extSoftVersion()["GDAL"])) >= "3.7.0")
file <- "etc/shapes/bhicv.gpkg.zip"
zipfile <- system.file(file, package="spdep")
if (GDAL37) {
    bh <- st_read(zipfile)
} else {
    td <- tempdir()
    bn <- sub(".zip", "", basename(file), fixed=TRUE)
    target <- unzip(zipfile, files=bn, exdir=td)
    bh <- st_read(target)
}
### data padronized
dpad <- data.frame(scale(as.data.frame(bh)[,5:8]))

### neighboorhod list
bh.nb <- poly2nb(bh)

### calculing costs
lcosts <- nbcosts(bh.nb, dpad)

### making listw
nb.w <- nb2listw(bh.nb, lcosts, style="B")
```

```
### find a minimum spanning tree
system.time(mst.bh <- mstree(nb.w,5))
dim(mst.bh)
head(mst.bh)
tail(mst.bh)
### the mstree plot
par(mar=c(0,0,0,0))
plot(st_geometry(bh), border=gray(.5))
plot(mst.bh, st_coordinates(st_centroid(bh)), col=2,
     cex.lab=.6, cex.circles=0.035, fg="blue", add=TRUE)
```

`nb.set.operations` *Set operations on neighborhood objects*

Description

Set operations on neighbors list objects

Usage

```
intersect.nb(nb.obj1,nb.obj2)
union.nb(nb.obj1,nb.obj2)
setdiff.nb(nb.obj1,nb.obj2)
complement.nb(nb.obj)
```

Arguments

`nb.obj`	a neighbor list created from any of the neighborhood list funtions
`nb.obj1`	a neighbor list created from any of the neighborhood list funtions
`nb.obj2`	a neighbor list created from any of the neighborhood list funtions

Details

These functions perform set operations on each element of a neighborlist. The arguments must be neighbor lists created from the same coordinates, and the region.id attributes must be identical.

Value

`nb.obj`	A new neighborlist created from the set operations on the input neighbor list(s)

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

`intersect.nb`, `union.nb`, `setdiff.nb`

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
col.tri.nb <- tri2nb(coords)
oldpar <- par(mfrow=c(1,2))
if (require("dbscan", quietly=TRUE)) {
  col.soi.nb <- graph2nb(soi.graph(col.tri.nb, coords))
  plot(st_geometry(columbus), border="grey")
  plot(col.soi.nb, coords, add=TRUE)
  title(main="Sphere of Influence Graph", cex.main=0.7)
  plot(st_geometry(columbus), border="grey")
  plot(complement.nb(col.soi.nb), coords, add=TRUE)
  title(main="Complement of Sphere of Influence Graph", cex.main=0.7)
}
par(mfrow=c(2,2))
col2 <- droplinks(col.gal.nb, 21)
plot(intersect.nb(col.gal.nb, col2), coords)
title(main="Intersect", cex.main=0.7)
plot(union.nb(col.gal.nb, col2), coords)
title(main="Union", cex.main=0.7)
plot(setdiff.nb(col.gal.nb, col2), coords)
title(main="Set diff", cex.main=0.7)
par(oldpar)
```

---

nb2blocknb *Block up neighbour list for location-less observations*

Description

The function blocks up a neighbour list for known spatial locations to create a new neighbour list for multiple location-less observations know to belong to the spatial locations, using the identification tags of the locations as the key.

Usage

```
nb2blocknb(nb=NULL, ID, row.names = NULL)
```

Arguments

nb	an object of class nb with a list of integer vectors containing neighbour region number ids; if null, an nb object with no neighbours is created the length of unique(as.character(ID))
ID	identification tags of the locations for the location-less observations; sort(unique(as.character(ID))) must be identical to sort(as.character(attr(nb, "region.id"))); same length as row.names if provided.
row.names	character vector of observation ids to be added to the neighbours list as attribute region.id, default seq(1, nrow(x)); same length as ID if provided.

Details

Assume that there is a list of unique locations, then a neighbour list can build for that, to create an input neighbour list. This needs to be "unfolded", so that observations belonging to each unique location are observation neighbours, and observations belonging to the location neighbours of the unique location in question are also observation neighbours, finally removing the observation itself (because it should not be its own neighbour). This scenario also arises when say only post codes are available, and some post codes contain multiple observations, where all that is known is that they belong to a specific post code, not where they are located within it (given that the post code locations are known).

Value

The function returns an object of class nb with a list of integer vectors containing neighbour observation number ids.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knn2nb, dnearneigh, cell2nb, tri2nb, poly2nb

Examples

```
## Not run:
data(boston, package="spData")
summary(as.vector(table(boston.c$TOWN)))
townaggr <- aggregate(boston.utm, list(town=boston.c$TOWN), mean)
block.rel <- graph2nb(relativeneigh(as.matrix(townaggr[,2:3])),
 as.character(townaggr[,1]), sym=TRUE)
block.rel
print(is.symmetric.nb(block.rel))
plot(block.rel, as.matrix(townaggr[,2:3]))
points(boston.utm, pch=18, col="lightgreen")
block.nb <- nb2blocknb(block.rel, as.character(boston.c$TOWN))
block.nb
print(is.symmetric.nb(block.nb))
plot(block.nb, boston.utm)
points(boston.utm, pch=18, col="lightgreen")
n.comp.nb(block.nb)$nc
moran.test(boston.c$CMEDV, nb2listw(boston.soi))
moran.test(boston.c$CMEDV, nb2listw(block.nb))
block.nb <- nb2blocknb(NULL, as.character(boston.c$TOWN))
block.nb
print(is.symmetric.nb(block.nb))
plot(block.nb, boston.utm)
n.comp.nb(block.nb)$nc
moran.test(boston.c$CMEDV, nb2listw(block.nb, zero.policy=TRUE), zero.policy=TRUE)

## End(Not run)
```

nb2INLA *Output spatial neighbours for INLA*

Description

Output spatial neighbours for INLA

Usage

```
nb2INLA(file, nb)
```

Arguments

file	file where adjacency matrix will be stored
nb	an object of class nb

Value

Nothing is returned but a file will be created with the representation of the adjacency matrix as required by INLA for its spatial models.

Author(s)

Virgilio Gomez-Rubio

References

http://www.r-inla.org

Examples

```
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
td <- tempdir()
x <- nb2INLA(paste(td, "columbus-INLA.adj", sep="/"), col.gal.nb)
readLines(paste(td, "columbus-INLA.adj", sep="/"), n=10)
```

nb2lines *Use vector files for import and export of weights*

Description

Use vector files for import and export of weights, storing spatial entity coordinates in the arcs, and the entity indices in the data frame.

Usage

```
nb2lines(nb, wts, coords, proj4string=NULL, as_sf=FALSE)
listw2lines(listw, coords, proj4string=NULL, as_sf=FALSE)
df2sn(df, i="i", i_ID="i_ID", j="j", wt="wt")
```

Arguments

nb	a neighbour object of class nb
wts	list of general weights corresponding to neighbours
coords	matrix of region point coordinates, a Spatial object (points or polygons), or an sfc object (points or polygons)
proj4string	default NULL; if coords is a Spatial or sf object, this value will be used, otherwise the value will be converted appropriately
as_sf	output object in Spatial or sf format, default FALSE, set to TRUE if coords is an sfc object and FALSE if a Spatial object
listw	a listw object of spatial weights
df	a data frame read from a shapefile, derived from the output of nb2lines
i	character name of column in df with from entity index
i_ID	character name of column in df with from entity region ID
j	character name of column in df with to entity index
wt	character name of column in df with weights

Details

The neighbour and weights objects may be retrieved by converting the specified columns of the data slot of the SpatialLinesDataFrame object into a spatial.neighbour object, which is then converted into a weights list object.

Value

nb2lines and listw2lines return a SpatialLinesDataFrame object or an sf object; the data frame contains with the from and to indices of the neighbour links and their weights. df2sn converts the data retrieved from reading the data from df back into a spatial.neighbour object.

Note

Original idea due to Gidske Leknes Andersen, Department of Biology, University of Bergen, Norway

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

sn2listw

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
res <- listw2lines(nb2listw(col.gal.nb), st_geometry(columbus))
summary(res)
tf <- paste0(tempfile(), ".gpkg")
st_write(res, dsn=tf, driver="GPKG")
inMap <- st_read(tf)
summary(inMap)
diffnb(sn2listw(df2sn(as.data.frame(inMap)))$neighbours, col.gal.nb)
res1 <- listw2lines(nb2listw(col.gal.nb), as(columbus, "Spatial"))
summary(res1)
```

---

nb2listw *Spatial weights for neighbours lists*

Description

The nb2listw function supplements a neighbours list with spatial weights for the chosen coding scheme. The can.be.simmed helper function checks whether a spatial weights object is similar to symmetric and can be so transformed to yield real eigenvalues or for Cholesky decomposition. The helper function listw2U() constructs a weights list object corresponding to the sparse matrix $\frac{1}{2}(\mathbf{W} + \mathbf{W}'$.

Usage

```
nb2listw(neighbours, glist=NULL, style="W", zero.policy=NULL)
listw2U(listw)
```

Arguments

neighbours	an object of class nb
glist	list of general weights corresponding to neighbours
style	style can take values “W”, “B”, “C”, “U”, “minmax” and “S”

zero.policy	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
listw	a listw object created for example by nb2listw

Details

Starting from a binary neighbours list, in which regions are either listed as neighbours or are absent (thus not in the set of neighbours for some definition), the function adds a weights list with values given by the coding scheme style chosen. B is the basic binary coding, W is row standardised (sums over all links to n), C is globally standardised (sums over all links to n), U is equal to C divided by the number of neighbours (sums over all links to unity), while S is the variance-stabilizing coding scheme proposed by Tiefelsdorf et al. 1999, p. 167-168 (sums over all links to n).

If zero policy is set to TRUE, weights vectors of zero length are inserted for regions without neighbour in the neighbours list. These will in turn generate lag values of zero, equivalent to the sum of products of the zero row `t(rep(0, length=length(neighbours)))` `%*%` `x`, for arbitraty numerical vector `x` of length `length(neighbours)`. The spatially lagged value of x for the zero-neighbour region will then be zero, which may (or may not) be a sensible choice.

If the sum of the glist vector for one or more observations is zero, a warning message is issued. The consequence for later operations will be the same as if no-neighbour observations were present and the zero.policy argument set to true.

The "minmax" style is based on Kelejian and Prucha (2010), and divides the weights by the minimum of the maximum row sums and maximum column sums of the input weights. It is similar to the C and U styles; it is also available in Stata.

Value

A listw object with the following members:

style	one of W, B, C, U, S, minmax as above
neighbours	the input neighbours list
weights	the weights for the neighbours and chosen style, with attributes set to report the type of relationships (binary or general, if general the form of the glist argument), and style as above

and attributes:

region.id	character, as the neighbour object
call	the function call
zero.policy	logical; the value of zero.policy when the object was created

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Tiefelsdorf, M., Griffith, D. A., Boots, B. 1999 A variance-stabilizing coding scheme for spatial link matrices, Environment and Planning A, 31, pp. 165–180; Kelejian, H. H., and I. R. Prucha. 2010. Specification and estimation of spatial autoregressive models with autoregressive and heteroskedastic disturbances. Journal of Econometrics, 157: pp. 53–67.

See Also

summary.nb, read.gal

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
cards <- card(col.gal.nb)
col.w <- nb2listw(col.gal.nb)
plot(cards, unlist(lapply(col.w$weights, sum)),xlim=c(0,10),
ylim=c(0,10), xlab="number of links", ylab="row sums of weights")
col.b <- nb2listw(col.gal.nb, style="B")
points(cards, unlist(lapply(col.b$weights, sum)), col="red")
col.c <- nb2listw(col.gal.nb, style="C")
points(cards, unlist(lapply(col.c$weights, sum)), col="green")
col.u <- nb2listw(col.gal.nb, style="U")
points(cards, unlist(lapply(col.u$weights, sum)), col="orange")
col.s <- nb2listw(col.gal.nb, style="S")
points(cards, unlist(lapply(col.s$weights, sum)), col="blue")
legend(x=c(0, 1), y=c(7, 9), legend=c("W", "B", "C", "U", "S"), bty="n",
col=c("black", "red", "green", "orange", "blue"), pch=rep(1,5), cex=0.8,
y.intersp=2.5)
summary(nb2listw(col.gal.nb, style="minmax"))
dlist <- nbdists(col.gal.nb, coords)
dlist <- lapply(dlist, function(x) 1/x)
col.w.d <- nb2listw(col.gal.nb, glist=dlist)
summary(unlist(col.w$weights))
summary(unlist(col.w.d$weights))
# introducing other conditions into weights - only earlier sales count
# see http://sal.uiuc.edu/pipermail/openspace/2005-October/000610.html
data(baltimore, package="spData")
set.seed(211)
dates <- sample(1:500, nrow(baltimore), replace=TRUE)
nb_15nn <- knn2nb(knearneigh(cbind(baltimore$X, baltimore$Y), k=15))
glist <- vector(mode="list", length=length(nb_15nn))
for (i in seq(along=nb_15nn))
  glist[[i]] <- ifelse(dates[i] > dates[nb_15nn[[i]]], 1, 0)
listw_15nn_dates <- nb2listw(nb_15nn, glist=glist, style="B")
which(lag(listw_15nn_dates, baltimore$PRICE) == 0.0)
which(sapply(glist, sum) == 0)
ex <- which(sapply(glist, sum) == 0)[1]
dates[ex]
dates[nb_15nn[[ex]]]
```

nb2listwdist *Distance-based spatial weights for neighbours lists*

Description

The nb2listwdist function supplements a neighbours list with spatial weights for the chosen types of distance modelling and coding scheme. While the offered coding schemes parallel those of the nb2listw function, three distance-based types of weights are available: inverse distance weighting (IDW), double-power distance weights, and exponential distance decay. The can.be.simmed helper function checks whether a spatial weights object is similar to symmetric and can be so transformed to yield real eigenvalues or for Cholesky decomposition.

Usage

```
nb2listwdist(neighbours, x, type="idw", style="raw",
  alpha = 1, dmax = NULL, longlat = NULL, zero.policy=NULL)
```

Arguments

neighbours	an object of class nb
x	an sp sf, or sfc object
type	default "idw"; the intended type of distance modelling, can take values "idw", "exp", and "dpd"
style	default "raw"; style can take values "raw", "W", "B", "C", "U", "minmax", and "S"
alpha	default 0; a parameter for controlling the distance modelling, see "Details"
dmax	default NULL, maximum distance threshold that is required for type "dpd" but optional for all other types
longlat	default NULL; TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in metres; if x is a SpatialPoints object, the value is taken from the object itself, and overrides this argument if not NULL; distances are measured in map units if FALSE or NULL
zero.policy	default NULL; use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors

Details

Starting from a binary neighbours list, in which regions are either listed as neighbours or are absent (thus not in the set of neighbours for some definition), the function adds a distance-based weights list. Three types of distance weight calculations based on pairwise distances d_{ij} are possible, all of which are controlled by parameter "alpha" (α below):

$$\text{idw: } w_{ij} = d_{ij}^{-\alpha},$$

$$\text{exp: } w_{ij} = \exp(-\alpha \cdot d_{ij}),$$

$$\text{dpd: } w_{ij} = \left[1 - (d_{ij}/d_{\max})^{\alpha}\right]^{\alpha},$$

the latter of which leads to $w_{ij} = 0$ for all $d_{ij} > d_{\max}$. Note that IDW weights show extreme behaviour close to 0 and can take on the value infinity. In such cases, the infinite values are replaced by the largest finite weight present in the weights list.

The default coding scheme is "raw", which outputs the raw distance-based weights without applying any kind of normalisation. In addition, the same coding scheme styles that are also available in the `nb2listw` function can be chosen. B is the basic binary coding, W is row standardised (sums over all links to n), C is globally standardised (sums over all links to n), U is equal to C divided by the number of neighbours (sums over all links to unity), while S is the variance-stabilising coding scheme proposed by Tiefelsdorf et al. 1999, p. 167-168 (sums over all links to n). The "minmax" style is based on Kelejian and Prucha (2010), and divides the weights by the minimum of the maximum row sums and maximum column sums of the input weights. It is similar to the C and U styles; it is also available in Stata.

If zero.policy is set to TRUE, weights vectors of zero length are inserted for regions without neighbour in the neighbours list. These will in turn generate lag values of zero, equivalent to the sum of products of the zero row `t(rep(0, length=length(neighbours)))` `%*%` `x`, for arbitraty numerical vector `x` of length `length(neighbours)`. The spatially lagged value of x for the zero-neighbour region will then be zero, which may (or may not) be a sensible choice.

Value

A `listw` object with the following members:

`style`	one of W, B, C, U, S, minmax as above
`type`	one of idw, exp, dpd as above
`neighbours`	the input neighbours list
`weights`	the weights for the neighbours and chosen style, with attributes set to report the type of relationships (binary or general, if general the form of the glist argument), and style as above

Author(s)

Rene Westerholt <rene.westerholt@tu-dortmund.de>

References

Tiefelsdorf, M., Griffith, D. A., Boots, B. 1999 A variance-stabilizing coding scheme for spatial link matrices, Environment and Planning A, 31, pp. 165–180; Kelejian, H. H., and I. R. Prucha. 2010. Specification and estimation of spatial autoregressive models with autoregressive and heteroskedastic disturbances. Journal of Econometrics, 157: pp. 53–67.

See Also

`nb2listw`, `summary.nb`

Examples

```
# World examples
data(world, package="spData")
# neighbours on distance interval [0, 1000] kilometres
# suppressWarnings(st_crs(world) <- "+proj=longlat") # for older PROJ
pts <- st_centroid(st_transform(world, 3857))
nb_world <- dnearneigh(pts, 0, 1000000)
# Moran's I (life expectancy) with IDW with alpha = 2, no coding scheme
world_weights <- nb2listwdist(nb_world, as(pts, "Spatial"), type = "idw",
  alpha = 2, zero.policy = TRUE)
moran.test(world$lifeExp, world_weights, zero.policy = TRUE, na.action = na.pass)
## Not run:
# Moran's I (life expectancy) with IDW with alpha = 2, no coding scheme
world_weights <- nb2listwdist(nb_world, pts, type = "idw",
  alpha = 2, zero.policy = TRUE)
moran.test(world$lifeExp, world_weights, zero.policy = TRUE, na.action = na.pass)
# Moran's I (life expectancy), DPD, alpha = 2, dmax = 1000 km, no coding scheme
world_weights <- nb2listwdist(nb_world, pts, type = "dpd",
  dmax = 1000000, alpha = 2, zero.policy = TRUE)
moran.test(world$lifeExp, world_weights, zero.policy = TRUE, na.action = na.pass)
# Boston examples
data(boston, package="spData")
boston_coords <- data.frame(x = boston.utm[,1], y = boston.utm[,2])
boston.geoms <- st_as_sf(boston_coords, coords = c("x", "y"), remove = FALSE)
nb_boston <- dnearneigh(boston.geoms, 0, 3)
# Moran's I (crime) with exp weights with alpha = 2, no coding scheme
boston_weights <- nb2listwdist(nb_boston, boston.geoms, type = "exp", alpha = 2,
  style="raw", zero.policy = TRUE)
moran.test(boston.c$CRIM, boston_weights, zero.policy = TRUE, na.action = na.pass)
# Moran's I (crime) with idw weights with alpha = 2, coding scheme = W
boston_weights <- nb2listwdist(nb_boston, boston.geoms, type = "idw", alpha = 2,
  style="W", zero.policy = TRUE)
moran.test(boston.c$CRIM, boston_weights, zero.policy = TRUE, na.action = na.pass)

## End(Not run)
```

---

nb2mat *Spatial weights matrices for neighbours lists*

Description

The function generates a weights matrix for a neighbours list with spatial weights for the chosen coding scheme.

Usage

```
nb2mat(neighbours, glist=NULL, style="W", zero.policy=NULL)
listw2mat(listw)
```

Arguments

neighbours	an object of class nb
glist	list of general weights corresponding to neighbours
style	style can take values W, B, C, and S
zero.policy	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
listw	a listw object from for example nb2listw

Details

Starting from a binary neighbours list, in which regions are either listed as neighbours or are absent (thus not in the set of neighbours for some definition), the function creates an n by n weights matrix with values given by the coding scheme style chosen. B is the basic binary coding, W is row standardised, C is globally standardised, while S is the variance-stabilizing coding scheme proposed by Tiefelsdorf et al. 1999, p. 167-168.

The function leaves matrix rows as zero for any regions with zero neighbours fore zero.policy TRUE. These will in turn generate lag values of zero, equivalent to the sum of products of the zero row `t(rep(0, length=length(neighbours))) %*% x`, for arbitraty numerical vector `x` of length `length(neighbours)`. The spatially lagged value of x for the zero-neighbour region will then be zero, which may (or may not) be a sensible choice.

Value

An n by n matrix, where n=length(neighbours)

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Tiefelsdorf, M., Griffith, D. A., Boots, B. 1999 A variance-stabilizing coding scheme for spatial link matrices, Environment and Planning A, 31, pp. 165-180.

See Also

nb2listw

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col005 <- dnearneigh(st_coordinates(st_centroid(st_geometry(columbus),
 of_largest_polygon=TRUE)), 0, 0.5, as.character(columbus$NEIGNO))
summary(col005)
col005.w.mat <- nb2mat(col005, style="B", zero.policy=TRUE)
table(round(rowSums(col005.w.mat)))
```

nb2WB *Output spatial weights for WinBUGS*

Description

Output spatial weights for WinBUGS

Usage

```
nb2WB(nb)
listw2WB(listw)
```

Arguments

nb	an object of class nb
listw	a listw object from for example nb2listw

Value

A list suitable for convering using dput for WinBUGS

Author(s)

Virgilio Gomez-Rubio

References

http://www.mrc-bsu.cam.ac.uk/bugs/winbugs/geobugs12manual.pdf

See Also

dput

Examples

```
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
x <- nb2WB(col.gal.nb)
dput(x, control=NULL)
x <- listw2WB(nb2listw(col.gal.nb))
dput(x, control=NULL)
```

nbcosts *Compute cost of edges*

Description

The cost of each edge is the distance between it nodes. This function compute this distance using a data.frame with observations vector in each node.

Usage

```
nbcost(data, id, id.neigh,  method = c("euclidean", "maximum",
    "manhattan", "canberra", "binary", "minkowski", "mahalanobis"),
    p = 2, cov, inverted = FALSE)
nbcosts(nb, data,  method = c("euclidean", "maximum",
    "manhattan", "canberra", "binary", "minkowski", "mahalanobis"),
    p = 2, cov, inverted = FALSE)
```

Arguments

nb	An object of nb class. See poly2nb for details.
data	A matrix with observations in the nodes.
id	Node index to compute the cost
id.neigh	Idex of neighbours nodes of node id
method	Character or function to declare distance method. If method is character, method must be "mahalanobis" or "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk". If method is one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk", see dist for details, because this function as used to compute the distance. If method="mahalanobis", the mahalanobis distance is computed between neighbour areas. If method is a function, this function is used to compute the distance.
p	The power of the Minkowski distance.
cov	The covariance matrix used to compute the mahalanobis distance.
inverted	logical. If 'TRUE', 'cov' is supposed to contain the inverse of the covariance matrix.

Value

A object of nbdist class. See nbdists for details.

Note

The neighbours must be a connected graph.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as nbdists, nb2listw

---

nbdists *Spatial link distance measures*

Description

Given a list of spatial neighbour links (a neighbours list of object type nb), the function returns the Euclidean distances along the links in a list of the same form as the neighbours list. If longlat = TRUE, Great Circle distances are used.

Usage

```
nbdists(nb, coords, longlat = NULL)
```

Arguments

nb	an object of class nb
coords	matrix of point coordinates, an object inheriting from SpatialPoints or an "sf" or "sfc" object; if the "sf" or "sfc" object geometries are in geographical coordinates (sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE), **s2** will be used to find distances https://github.com/r-spatial/s2/issues/125
longlat	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if coords is a SpatialPoints object, the value is taken from the object itself

Value

A list with class nbdist

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb, nb2listw

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
dlist <- nbdists(col.gal.nb, coords)
dlist <- lapply(dlist, function(x) 1/x)
stem(unlist(dlist))
```

nblag *Higher order neighbours lists*

Description

The function creates higher order neighbour lists, where higher order neighbours are only lags links from each other on the graph described by the input neighbours list. It will refuse to lag neighbours lists with the attribute self.included set to TRUE. nblag_cumul cumulates neighbour lists to a single neighbour list ("nb" object).

Usage

```
nblag(neighbours, maxlag)
nblag_cumul(nblags)
```

Arguments

neighbours	input neighbours list of class nb
maxlag	the maximum lag to be constructed
nblags	a list of neighbour lists as output by nblag

Value

returns a list of lagged neighbours lists each with class nb

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> and Giovanni Millo

See Also

summary.nb

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
summary(col.gal.nb, coords)
col.lags <- nblag(col.gal.nb, 2)
print(col.lags)
summary(col.lags[[2]], coords)
plot(st_geometry(columbus), border="grey")
plot(col.gal.nb, coords, add=TRUE)
title(main="GAL order 1 (black) and 2 (red) links")
plot(col.lags[[2]], coords, add=TRUE, col="red", lty=2)
cuml <- nblag_cumul(col.lags)
cuml
```

```
run <- FALSE
if (require(igraph, quietly=TRUE) && require(spatialreg, quietly=TRUE)) run <- TRUE
if (run) {
W <- as(nb2listw(col.gal.nb), "CsparseMatrix")
G <- graph_from_adjacency_matrix(W, mode="directed", weight="W")
D <- diameter(G)
nbs <- nblag(col.gal.nb, maxlag=D)
n <- length(col.gal.nb)
lmat <- lapply(nbs, nb2mat, style="B", zero.policy=TRUE)
mat <- matrix(0, n, n)
for (i in seq(along=lmat)) mat = mat + i*lmat[[i]]
G2 <- distances(G)
print(all.equal(G2, mat, check.attributes=FALSE))
}
```

`oldcol` *Columbus OH spatial analysis data set - old numbering*

Description

The `COL.OLD` data frame has 49 rows and 22 columns. The observations are ordered and numbered as in the original analyses of the data set in the SpaceStat documentation and in Anselin, L. 1988 Spatial econometrics: methods and models, Dordrecht: Kluwer. Unit of analysis: 49 neighbourhoods in Columbus, OH, 1980 data. In addition the data set includes `COL.nb`, the neighbours list as used in Anselin (1988).

Usage

```
data(oldcol)
```

Format

This data frame contains the following columns:

AREA_PL computed by ArcView (agrees with areas of polygons in the "columbus" data set

PERIMETER computed by ArcView

COLUMBUS. internal polygon ID (ignore)

COLUMBUS.I another internal polygon ID (ignore)

POLYID yet another polygon ID

NEIG neighborhood id value (1-49); conforms to id value used in Spatial Econometrics book.

HOVAL housing value (in $1,000)

INC household income (in $1,000)

CRIME residential burglaries and vehicle thefts per thousand households in the neighborhood

OPEN open space in neighborhood

PLUMB percentage housing units without plumbin

DISCBD distance to CBD

X x coordinate (in arbitrary digitizing units, not polygon coordinates)

Y y coordinate (in arbitrary digitizing units, not polygon coordinates)

AREA_SS neighborhood area (computed by SpaceStat)

NSA north-south dummy (North=1)

NSB north-south dummy (North=1)

EW east-west dummy (East=1)

CP core-periphery dummy (Core=1)

THOUS constant=1,000

NEIGNO NEIG+1,000, alternative neighborhood id value

PERIM polygon perimeter (computed by SpaceStat)

Details

The row names of `COL.OLD` and the `region.id` attribute of `COL.nb` are set to `columbus$NEIGNO`.

Note

All source data files prepared by Luc Anselin, Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign.

Source

Anselin, Luc. 1988. Spatial econometrics: methods and models. Dordrecht: Kluwer Academic, Table 12.1 p. 189.

---

p.adjustSP *Adjust local association measures' p-values*

Description

Make an adjustment to local association measures' p-values based on the number of neighbours (+1) of each region, rather than the total number of regions.

Usage

```
p.adjustSP(p, nb, method = "none")
```

Arguments

p	vector of p-values
nb	a list of neighbours of class nb
method	correction method as defined in p.adjust: "The adjustment methods include the Bonferroni correction ('"bonferroni"') in which the p-values are multiplied by the number of comparisons. Four less conservative corrections are also included by Holm (1979) ('holm'), Hochberg (1988) ('hochberg'), Hommel (1988) ('hommel') and Benjamini & Hochberg (1995) ('fdr'), respectively. A pass-through option ('none') is also included."

Value

A vector of corrected p-values using only the number of neighbours + 1.

Author(s)

Danlin Yu and Roger Bivand <Roger.Bivand@nhh.no>

See Also

p.adjust, localG, localmoran

Examples

```
data(afcon, package="spData")
oid <- order(afcon$id)
resG <- as.vector(localG(afcon$totcon, nb2listw(include.self(paper.nb))))
non <- format.pval(pnorm(2*(abs(resG)), lower.tail=FALSE), 2)
bon <- format.pval(p.adjustSP(pnorm(2*(abs(resG)), lower.tail=FALSE),
 paper.nb, "bonferroni"), 2)
tot <- format.pval(p.adjust(pnorm(2*(abs(resG)), lower.tail=FALSE),
 "bonferroni", n=length(resG)), 2)
data.frame(resG, non, bon, tot, row.names=afcon$name)[oid,]
```

---

plot.mst *Plot the Minimum Spanning Tree*

Description

This function plots a MST, the nodes are circles and the edges are segments.

Usage

```
## S3 method for class 'mst'
plot(x, coords, label.areas = NULL,
   cex.circles = 1, cex.labels = 1, add=FALSE, ...)
```

Arguments

x	Object of mst class.
coords	A two column matrix with the coordinates of nodes.
label.areas	A vector with the labels of nodes
cex.circles	The length of circles to plot.
cex.labels	The length of nodes labels ploted.
add	default FALSE, create new plot
...	Further arguments passed to plotting functions.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as skater and mstree

Examples

```
### see example in mstree function documentation
```

---

plot.nb *Plot a neighbours list*

Description

A function to plot a neighbours list given point coordinates to represent the region in two dimensions; plot.listw is a wrapper that passes its neighbours component to plot.nb.

Usage

```
## S3 method for class 'nb'
plot(x, coords, col="black", points=TRUE, add=FALSE, arrows=FALSE,
 length=0.1, xlim=NULL, ylim=NULL, ...)
## S3 method for class 'listw'
plot(x, coords, col="black", points=TRUE, add=FALSE, arrows=FALSE,
 length=0.1, xlim=NULL, ylim=NULL, ...)
```

Arguments

x	an object of class nb or (for plot.listw) class listw
coords	matrix of region point coordinates, a Spatial object (points or polygons), or an sfc object (points or polygons)
col	plotting colour
points	(logical) add points to plot

add	(logical) add to existing plot
arrows	(logical) draw arrowheads for asymmetric neighbours
length	length in plot inches of arrow heads drawn for asymmetric neighbours lists
xlim, ylim	plot window bounds
...	further graphical parameters as in par(..)

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
plot(col.gal.nb, st_geometry(columbus))
title(main="GAL order 1 links with first nearest neighbours in red", cex.main=0.6)
plot(col.gal.nb, as(columbus, "Spatial"))
title(main="GAL order 1 links with first nearest neighbours in red", cex.main=0.6)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
col.knn <- knearneigh(coords, k=1)
plot(knn2nb(col.knn), coords, add=TRUE, col="red", length=0.08)
```

---

plot.skater *Plot the object of skater class*

Description

This function displays the results of the skater function. The subgraphs are plotted with different colours.

Usage

```
## S3 method for class 'skater'
plot(x, coords, label.areas = NULL,
   groups.colors, cex.circles = 1, cex.labels = 1, ...)
```

Arguments

x	An object of skater class.
coords	A matrix of two colums with coordinates of nodes.
label.areas	A vector of labels of nodes.
groups.colors	A vector with colors of groups ou sub-graphs.

cex.circles	The length of circles with represent the nodes.
cex.labels	The length of labels of nodes.
...	Further arguments passed to plotting funcitons.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as skater and mstree

Examples

```
### see example in the skater function documentation
```

---

poly2nb *Construct neighbours list from polygon list*

Description

The function builds a neighbours list based on regions with contiguous boundaries, that is sharing one or more boundary point. The current function is in part interpreted and may run slowly for many regions or detailed boundaries, but from 0.2-16 should not fail because of lack of memory when single polygons are built of very many border coordinates.

Usage

```
poly2nb(pl, row.names = NULL, snap=NULL, queen=TRUE, useC=TRUE, foundInBox=NULL)
```

Arguments

pl	list of polygons of class extending SpatialPolygons, or an sf or sfc object containing non-empty (multi-)polygon objects
row.names	character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1, nrow(x)); if pl has row.names, they are used instead of the default sequence.
snap	boundary points less than snap distance apart are considered to indicate contiguity; used both to find candidate and actual neighbours for planar geometries, but only actual neighbours for spherical geometries, as spherical spatial indexing itself injects some fuzzyness. If not set, for all SpatialPolygons objects, the default is as before sqrt(.Machine$double.eps), with this value also used for sf objects with no coordinate reference system. For sf objects with a defined coordinate reference system, the default value is 1e-7 for geographical coordinates (approximately 10mm), is 10mm where projected coordinates are in metre units, and is converted from 10mm to the same distance in the units of the coordinates. Should the conversion fail, snap reverts to sqrt(.Machine$double.eps).

queen — if TRUE, a single shared boundary point meets the contiguity condition, if FALSE, more than one shared point is required; note that more than one shared boundary point does not necessarily mean a shared boundary line

useC — default TRUE, doing the work loop in C, may be set to false to revert to R code calling two C functions in an n*k work loop, where k is the average number of candidate neighbours

foundInBox — default NULL using R code or st_intersects() to generate candidate neighbours (using snap= if the geometries are not spherical); if not NULL (for legacy purposes) a list of length (n-1) with integer vectors of candidate neighbours (j > i) (as created by the poly_findInBoxGEOS function in **rgeos** for clean polygons)

Value

A neighbours list with class nb. See card for details of "nb" objects.

Note

From 0.5-8, the function includes faster bounding box indexing and other improvements contributed by Micah Altman. If a cluster is provided using set.ClusterOption, it will be used for finding candidate bounding box overlaps for exact testing for contiguity.

Until 1.1-7, sf polygons included both start and end points, so could erroneously report queen neighbourhood where only rook was present, see https://github.com/r-spatial/spdep/issues/50.

From 1.1-9 with **sf** 1.0-0, **s2** is used in bounding box indexing internally when pl is in geographical coordinates. Because the topology engine of **s2** differs from the use of GEOS for planar coordinates by **sf**, some output differences may be expected. Since treating spherical geometries as planar is also questionable, it is not clear whether spherical contiguous polygon neighbours should simply follow neighbours found by treating the geometries as planar https://github.com/r-spatial/s2/issues/125#issuecomment-864403372. However, current advice is not necessarily to use **s2** for finding contiguity neighbours, or at least to check output.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> with contributions from Micah Altman

See Also

summary.nb, card

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(st_geometry(columbus)))
xx <- poly2nb(as(columbus, "Spatial"))
dxx <- diffnb(xx, col.gal.nb)
plot(st_geometry(columbus), border="grey")
plot(col.gal.nb, coords, add=TRUE)
```

```
plot(dxx, coords, add=TRUE, col="red")
title(main=paste("Differences (red) in Columbus GAL weights (black)",
 "and polygon generated queen weights", sep="\n"), cex.main=0.6)
# poly2nb with sf sfc_MULTIPOLYGON objects
sf_xx <- poly2nb(columbus)
diffnb(sf_xx, xx)
sfc_xx <- poly2nb(st_geometry(columbus))
diffnb(sfc_xx, xx)
xxx <- poly2nb(as(columbus, "Spatial"), queen=FALSE)
dxxx <- diffnb(xxx, col.gal.nb)
plot(st_geometry(columbus), border = "grey")
plot(col.gal.nb, coords, add = TRUE)
plot(dxxx, coords, add = TRUE, col = "red")
title(main=paste("Differences (red) in Columbus GAL weights (black)",
 "and polygon generated rook weights", sep="\n"), cex.main=0.6)
cards <- card(xx)
maxconts <- which(cards == max(cards))
if(length(maxconts) > 1) maxconts <- maxconts[1]
fg <- rep("grey", length(cards))
fg[maxconts] <- "red"
fg[xx[[maxconts]]] <- "green"
plot(st_geometry(columbus), col=fg)
title(main="Region with largest number of contiguities", cex.main=0.6)
nc.sids <- st_read(system.file("shapes/sids.gpkg", package="spData")[1], quiet=TRUE)
system.time(xxnb <- poly2nb(nc.sids))
system.time(xxnb <- poly2nb(as(nc.sids, "Spatial")))
plot(st_geometry(nc.sids))
plot(xxnb, st_coordinates(st_centroid(nc.sids)), add=TRUE, col="blue")
sq <- st_polygon(list(rbind(c(0,0), c(1,0), c(1,1), c(0,1), c(0,0))))
sq2 <- sq + c(0,1)
sq3 <- sq + c(1,0)
sq4 <- sq + c(1,1)
gm <- st_sfc(list(sq, sq2, sq3, sq4))
df <- st_as_sf(data.frame(gm, id=1:4))
plot(st_geometry(df))
text(st_coordinates(st_centroid(gm)), as.character(df$id))
unclass(poly2nb(df, queen = FALSE))
col_geoms <- st_geometry(columbus)
col_geoms[1] <- st_buffer(col_geoms[1], dist=-0.05)
st_geometry(columbus) <- col_geoms
poly2nb(columbus)
```

---

probmap *Probability mapping for rates*

Description

The function returns a data frame of rates for counts in populations at risk with crude rates, expected counts of cases, relative risks, and Poisson probabilities.

Usage

```
probmap(n, x, row.names=NULL, alternative="less")
```

Arguments

n	a numeric vector of counts of cases
x	a numeric vector of populations at risk
row.names	row names passed through to output data frame
alternative	default "less", may be set to "greater"

Details

The function returns a data frame, from which rates may be mapped after class intervals have been chosen. The class intervals used in the examples are mostly taken from the referenced source.

Value

raw	raw (crude) rates
expCount	expected counts of cases assuming global rate
relRisk	relative risks: ratio of observed and expected counts of cases multiplied by 100
pmap	Poisson probability map values: probablility of getting a more "extreme" count than actually observed - one-tailed, default alternative observed "less" than expected

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Bailey T, Gatrell A (1995) Interactive Spatial Data Analysis, Harlow: Longman, pp. 300–303.

See Also

EBest, EBlocal, ppois

Examples

```
auckland <- st_read(system.file("shapes/auckland.gpkg", package="spData")[1], quiet=TRUE)
res <- probmap(auckland$M77_85, 9*auckland$Und5_81)
rt <- sum(auckland$M77_85)/sum(9*auckland$Und5_81)
ppois_pmap <- numeric(length(auckland$Und5_81))
for (i in seq(along=ppois_pmap)) {
ppois_pmap[i] <- poisson.test(auckland$M77_85[i], r=rt,
  T=(9*auckland$Und5_81[i]), alternative="less")$p.value
all.equal(ppois_pmap, res$pmap)
}
res$id <- 1:nrow(res)
```

```
auckland$id <- res$id <- 1:nrow(res)
auckland_res <- merge(auckland, res, by="id")
plot(auckland_res[, "raw"], main="Crude (raw) estimates")
plot(auckland_res[, "relRisk"], main="Standardised mortality ratios")
plot(auckland_res[, "pmap"], main="Poisson probabilities",
 breaks=c(0, 0.05, 0.1, 0.5, 0.9, 0.95, 1))
```

`prunecost` *Compute cost of prune each edge*

Description

If any edge are dropped, the MST are pruned. This generate a two subgraphs. So, it makes a tree graphs and tree dissimilarity values are computed, one for each graph. The dissimilarity is the sum over sqared differences between the observactions in the nodes and mean vector of observations in the graph. The dissimilarity of original graph and the sum of dissimilarity of subgraphs are returned.

Usage

```
prunecost(edges, data, method = c("euclidean", "maximum", "manhattan",
    "canberra", "binary", "minkowski", "mahalanobis"),
    p = 2, cov, inverted = FALSE)
```

Arguments

`edges`	A matrix with 2 colums with each row is one edge
`data`	A data.frame with observations in the nodes.
`method`	Character or function to declare distance method. If `method` is character, method must be "mahalanobis" or "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk". If `method` is one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk", see `dist` for details, because this function as used to compute the distance. If `method="mahalanobis"`, the mahalanobis distance is computed between neighbour areas. If `method` is a `function`, this function is used to compute the distance.
`p`	The power of the Minkowski distance.
`cov`	The covariance matrix used to compute the mahalanobis distance.
`inverted`	logical. If 'TRUE', 'cov' is supposed to contain the inverse of the covariance matrix.

Value

A vector with the differences between the dissimilarity of all nodes and the dissimilarity sum of all subgraphs obtained by pruning one edge each time.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as prunemst

Examples

```
d <- data.frame(a=-2:2, b=runif(5))
e <- matrix(c(1,2, 2,3, 3,4, 4,5), ncol=2, byrow=TRUE)

sum(sweep(d, 2, colMeans(d))^2)

prunecost(e, d)
```

---

prunemst *Prune a Minimun Spanning Tree*

Description

This function deletes a first edge and makes two subsets of edges. Each subset is a Minimun Spanning Treee.

Usage

```
prunemst(edges, only.nodes = TRUE)
```

Arguments

edges	A matrix with two colums with each row is one edge
only.nodes	If only.nodes=FALSE, return a edges and nodes of each MST resulted. If only.nodes=TRUE, return a two sets of nodes. Defalt is TRUE

Value

A list of length two. If only.nodes=TRUE each element is a vector of nodes. If only.nodes=FALSE each element is a list with nodes and edges.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as mstree

Examples

```
e <- matrix(c(2,3, 1,2, 3,4, 4,5), ncol=2, byrow=TRUE)
e
prunemst(e)
prunemst(e, only.nodes=FALSE)
```

read.gal *Read a GAL lattice file into a neighbours list*

Description

The function read.gal() reads a GAL lattice file into a neighbours list for spatial analysis. It will read old and new style (GeoDa) GAL files. The function read.geoda is a helper file for reading comma separated value data files, calling read.csv().

Usage

```
read.gal(file, region.id=NULL, override.id=FALSE)
read.geoda(file, row.names=NULL, skip=0)
```

Arguments

file	name of file with GAL lattice data
region.id	region IDs in specified order to coerse neighbours list order and numbering to that of the region.id
override.id	override any given (or NULL) region.id, collecting region.id numbering and order from the GAL file.
row.names	as in row.names in read.csv(), typically a character string naming the column of the file to be used
skip	skip number of lines, as in read.csv()

Details

Luc Anselin (2003): Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, now dead link: http://www.csiss.org/gispopsci/workshops/2011/PSU/read
Luc Anselin (2003) *GeoDa 0.9 User's Guide*, pp. 80–81, Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, http://geodacenter.github.io/docs/geoda093.pdf; GAL - Geographical Algorithms Library, University of Newcastle

Value

The function read.gal() returns an object of class nb with a list of integer vectors containing neighbour region number ids. The function read.geoda returns a data frame, and issues a warning if the returned object has only one column.

Note

Example data originally downloaded from now dead link: http://sal.agecon.uiuc.edu/weights/zips/us48.zip

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb

Examples

```
us48.fipsno <- read.geoda(system.file("etc/weights/us48.txt",
 package="spdep")[1])
us48.q <- read.gal(system.file("etc/weights/us48_q.GAL", package="spdep")[1],
 us48.fipsno$Fipsno)
us48.r <- read.gal(system.file("etc/weights/us48_rk.GAL", package="spdep")[1],
 us48.fipsno$Fipsno)
data(state)
if (as.numeric(paste(version$major, version$minor, sep="")) < 19) {
 m50.48 <- match(us48.fipsno$"State.name", state.name)
} else {
 m50.48 <- match(us48.fipsno$"State_name", state.name)
}
plot(us48.q, as.matrix(as.data.frame(state.center))[m50.48,])
plot(diffnb(us48.r, us48.q),
 as.matrix(as.data.frame(state.center))[m50.48,], add=TRUE, col="red")
title(main="Differences between rook and queen criteria imported neighbours lists")
```

---

read.gwt2nb *Read and write spatial neighbour files*

Description

The "gwt" functions read and write GeoDa GWT files (the example file baltk4.GWT was downloaded from the site given in the reference), and the "dat" functions read and write Matlab sparse matrix files as used by James LeSage's Spatial Econometrics Toolbox (the example file wmat.dat was downloaded from the site given in the reference). The body of the files after any headers should have three columns separated by white space, and the third column must be numeric in the locale of the reading platform (correct decimal separator).

Usage

```
read.gwt2nb(file, region.id=NULL)
write.sn2gwt(sn, file, shpfile=NULL, ind=NULL, useInd=FALSE, legacy=FALSE)
read.dat2listw(file)
write.sn2dat(sn, file)
read.swmdbf2listw(fn, region.id=NULL, style=NULL, zero.policy=NULL)
read_swm_dbf(fn)
write.sn2DBF(sn, file)
```

Arguments

file, fn	name of file with weights data

region.id	a character vector of region IDs - for ArcGIS SWM DBFs, the values must be character integers (only numbers not starting with zero)
sn	a spatial.neighbour object
shpfile	character string: if not given Shapefile name taken from GWT file for this dataset
ind	character string: region id indicator field name
useInd	default FALSE, if TRUE, write region.id attribute ID key tags to output file (use in OpenGeoDa will depend on the shapefile having the field named in the ind argument matching the exported tags)
legacy	default FALSE; if TRUE, header has single field with number of observations only
style	default NULL, missing, set to "M" and warning given; if not "M", passed to nb2listw to re-build the object
zero.policy	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors

Details

Attempts to honour the region.id argument given when reading GWT and SWM/DBF files. If the region IDs given in region.id= do not match the origins or destinations in the GWT file, an error or warning will be thrown, which should be considered carefully. read_swm_dbf is a simplified interface to read.swmdbf2listw When no-neighbour observations are present, it is advised that region.id= be given in read.swmdbf2listw; while the function should read correctly if the minimum and maximum IDs are present as observations with neighbours without region.id= set, reading using read.swmdbf2listw will fail when the minimum or maximum ID observations have no neighbours and region.id= is not given.

Value

read.gwt2nb returns a neighbour "nb" object with the generalised weights stored as a list element called "dlist" of the "GeoDa" attribute; read.swmdbf2listw returns a "listw" object read from a DBF file exported from an ArcGIS SWM object.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Luc Anselin (2003) *GeoDa 0.9 User's Guide*, pp. 80–81, Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, http://geodacenter.github.io/docs/geoda093.pdf; also material formerly at spatial-econometrics.com/data/contents.html

See Also

read.gal

Examples

```
data(baltimore, package="spData")
STATION <- baltimore$STATION
gwt1 <- read.gwt2nb(system.file("weights/baltk4.GWT", package="spData")[1],
 STATION)
cat(paste("Neighbours list symmetry;", is.symmetric.nb(gwt1, FALSE, TRUE),
 "\n"))
listw1 <- nb2listw(gwt1, style="B", glist=attr(gwt1, "GeoDa")$dist)
tmpGWT <- tempfile()
write.sn2gwt(listw2sn(listw1), tmpGWT)
gwt2 <- read.gwt2nb(tmpGWT, STATION)
cat(paste("Neighbours list symmetry;", is.symmetric.nb(gwt2, FALSE, TRUE),
 "\n"))
diffnb(gwt1, gwt2)
data(oldcol)
tmpMAT <- tempfile()
COL.W <- nb2listw(COL.nb)
write.sn2dat(listw2sn(COL.W), tmpMAT)
listwmat1 <- read.dat2listw(tmpMAT)
diffnb(listwmat1$neighbours, COL.nb, verbose=TRUE)
listwmat2 <- read.dat2listw(system.file("etc/weights/wmat.dat",
 package="spdep")[1])
diffnb(listwmat1$neighbours, listwmat2$neighbours, verbose=TRUE)
if (require("foreign", quietly=TRUE)) {
nc_sf <- sf::st_read(system.file("gpkg/nc.gpkg", package="sf")[1])
nc_sf$UniqueID <- 1:nrow(nc_sf)
fn <- system.file("etc/misc/nc_contiguity_unique_id.dbf", package="spdep")[1]
nc1 <- read.swmdbf2listw(fn, style="B")
nc1
nc1a <- read.swmdbf2listw(fn, region.id=as.character(nc_sf$UniqueID),
 style="B")
all.equal(nc1, nc1a)
fn <- system.file("etc/misc/nc_contiguity_unique_id_islands.dbf",
 package="spdep")[1]
try(nc1i <- read.swmdbf2listw(fn, style="B"))
nc1i <- read.swmdbf2listw(fn, style="B", zero.policy=TRUE)
nc1ia <- read.swmdbf2listw(fn, region.id=as.character(nc_sf$UniqueID),
 style="B", zero.policy=TRUE)
nc1ia
all.equal(nc1i, nc1ia)
(GDAL37 <- as.numeric_version(unname(sf_extSoftVersion()["GDAL"])) >= "3.7.0")
file <- "etc/shapes/california.gpkg.zip"
zipfile <- system.file(file, package="spdep")
if (GDAL37) {
    cal <- st_read(zipfile)
} else {
    td <- tempdir()
    bn <- sub(".zip", "", basename(file), fixed=TRUE)
    target <- unzip(zipfile, files=bn, exdir=td)
    cal <- st_read(target)
}
fn <- system.file("etc/misc/contiguity_myid.dbf", package="spdep")[1]
```

```
cal1 <- read.swmdbf2listw(fn, style="B")
cal1a <- read.swmdbf2listw(fn, region.id=as.character(cal$MYID), style="B")
all.equal(cal1, cal1a)
fn <- system.file("etc/misc/contiguity_unique_id.dbf", package="spdep")[1]
cal2 <- read.swmdbf2listw(fn, style="B")
cal2a <- read.swmdbf2listw(fn, region.id=as.character(cal$UniqueID), style="B")
all.equal(cal2, cal2a)
fn <- system.file("etc/misc/contiguity_unique_id_islands.dbf", package="spdep")[1]
try(cal3i <- read.swmdbf2listw(fn, style="B"))
cal3i <- read.swmdbf2listw(fn, style="B", zero.policy=TRUE)
cal3ia <- read.swmdbf2listw(fn, region.id=as.character(cal$UniqueID), style="B", zero.policy=TRUE)
all.equal(cal3i, cal3ia)
cal1a_1n_nb <- cal1a$neighbours
cal1a_1n_nb <- droplinks(cal1a_1n_nb, drop=c("158", "180", "215"), sym=TRUE)
cal1a_1n <- nb2listw(cal1a_1n_nb, style="B", zero.policy=TRUE)
cal1a_1n_sn <- listw2sn(cal1a_1n)
file <- tempfile(fileext=".dbf")
write.sn2DBF(cal1a_1n_sn, file)
cal1a_1n_rt <- read.swmdbf2listw(file, region.id=as.character(cal$MYID),
 style="B", zero.policy=TRUE)
all.equal(cal1a_1n$neighbours, cal1a_1n_rt$neighbours)
all.equal(cal1a_1n$weights, cal1a_1n_rt$weights, check.attributes=FALSE)
cal1_1n_rt <- read.swmdbf2listw(file, style="B", zero.policy=TRUE)
all(isTRUE(all.equal(cal1a_1n$neighbours, cal1_1n_rt$neighbours)))
all(isTRUE(all.equal(cal1a_1n$weights, cal1_1n_rt$weights)))

}
```

`Rotation` *Rotate a set of point by a certain angle*

Description

Rotate a set of XY coordinates by an angle (in radians)

Usage

```
Rotation(xy, angle)
```

Arguments

`xy`	A 2-columns matrix or data frame containing a set of X and Y coordinates.
`angle`	Numeric. A scalar giving the angle at which the points should be rotated. The angle is in radians.

Value

A 2-columns matrix of the same size as `xy` giving the rotated coordinates.

Author(s)

F. Guillaume Blanchet

Examples

```
set.seed(1)
### Create a set of coordinates
coords <- cbind(runif(20), runif(20))

### Create a series of angles
rad <- seq(0, pi, l=20)

opar <- par(mfrow=c(5,4), mar=c(3,3,1,1))
for(i in rad){
coords.rot <- Rotation(coords, i)
plot(coords.rot, xlab="", ylab="")
}
par(opar)

### Rotate the coordinates by an angle of 90 degrees
coords.90 <- Rotation(coords, 90*pi/180)
coords.90

plot(coords, xlim=range(rbind(coords.90,coords)[,1]),
 ylim=range(rbind(coords.90,coords)[,2]), asp=1)
points(coords.90, pch=19)
```

SD.RStests *Rao's score and adjusted Rao's score tests of linear hypotheses for spatial Durbin and spatial Durbin error models*

Description

Rao's score and adjusted Rao's score tests of linear hypotheses applied to a fitted linear model to examine whether either the spatially lagged dependent variable `lag` or the spatially lagged independent variable(s) `WX` should be included in the model, or both (SDM). Adjusted tests are provided for `lag` and `WX` adapting to the presence of the other, and a joint test for both. The joint test is equal to the unadjusted of one plus the adjusted of the other. In addition, draft tests are added from Koley (2024, section 6) for spatial Durbin error models to examine whether either the spatially lagged error `err` or the spatially lagged independent variable(s) `WX` should be included in the model, or both (SDEM); because of orthogonality, no adjusted tests are required.

Usage

```
SD.RStests(model, listw, zero.policy = attr(listw, "zero.policy"), test = "SDM",
 Durbin = TRUE)
```

Arguments

model	an object of class lm returned by lm
listw	a listw object created for example by nb2listw, expected to be row-standardised (W-style)
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
test	test="SDM" computes the SDM tests, a character vector of tests requested chosen from SDM_RSlag, SDM_adjRSlag, SDM_RSWX, SDM_adjRSWX, SDM_Joint, test="SDEM" computes the SDEM tests, a character vector of tests requested chosen from SDEM_RSerr, SDEM_RSWX, SDEM_Joint; test="all" computes all the tests
Durbin	default TRUE for Durbin models including WX; if TRUE, full spatial Durbin model; if a formula object, the subset of explanatory variables to lag

Value

A list of class LMtestlist of htest objects, each with:

statistic	the value of the Lagrange Multiplier test.
parameter	number of degrees of freedom
p.value	the p-value of the test.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data.

Note

The results in the example below agree with those in Table 3, p. 22 in Koley and Bera (2024).

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>, Malabika Koley and Anil K. Bera

References

Malabika Koley and Anil K. Bera (2024) To use, or not to use the spatial Durbin model? – that is the question, Spatial Economic Analysis, 19:1, 30-56, doi:10.1080/17421772.2023.2256810; Malabika Koley (2024) Specification Testing under General Nesting Spatial Model (Appendix C), https://sites.google.com/view/malabikakoley/research.

See Also

lm, lm.RStests

Examples

```
columbus <- sf::st_read(system.file("shapes/columbus.gpkg", package="spData")[1])
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
col.listw <- nb2listw(col.gal.nb, style="W")
lm_obj <- lm(CRIME ~ INC + HOVAL, data=columbus)
summary(lm.RStests(lm_obj, col.listw, test="all"))
res <- SD.RStests(lm_obj, col.listw, test="SDM")
summary(res)
all.equal(unname(res$SDM_Joint$statistic),
 unname(res$SDM_RSlag$statistic + res$SDM_adjRSWX$statistic))
all.equal(unname(res$SDM_Joint$statistic),
 unname(res$SDM_adjRSlag$statistic + res$SDM_RSWX$statistic))
res <- SD.RStests(lm_obj, col.listw, test="SDEM")
summary(res)
all.equal(unname(res$SDEM_Joint$statistic),
 unname(res$SDEM_RSerr$statistic + res$SDEM_RSWX$statistic))
summary(SD.RStests(lm_obj, nb2listw(col.gal.nb, style="C"), test="all"))
summary(SD.RStests(lm_obj, col.listw, test="all", Durbin= ~ INC))
lm_obj0 <- lm(I(scale(CRIME)) ~ 0 + I(scale(INC)) + I(scale(HOVAL)),
 data=columbus)
summary(SD.RStests(lm_obj0, col.listw, test="all"))
columbusNA <- columbus
columbusNA$HOVAL[15] <- NA
lm_objNA <- lm(CRIME ~ INC + HOVAL, data=columbusNA)
summary(SD.RStests(lm_objNA, col.listw, test="all"))
```

---

set.mcOption *Options for parallel support*

Description

Provides support for the use of parallel computation in the parallel package.

Usage

```
set.mcOption(value)
get.mcOption()
set.coresOption(value)
get.coresOption()
set.ClusterOption(cl)
get.ClusterOption()
```

Arguments

value	valid replacement value
cl	a cluster object created by makeCluster in **parallel**

Details

Options in the spdep package are held in an environment local to the package namespace and not exported. Option values are set and retrieved with pairs of access functions, get and set. The `mc` option is set by default to FALSE on Windows systems, as they cannot fork the R session; by default it is TRUE on other systems, but may be set FALSE. If `mc` is FALSE, the `Cluster` option is used: if `mc` is FALSE and the `Cluster` option is NULL no parallel computing is done, or the `Cluster` option is passed a "cluster" object created by the parallel or snow package for access without being passed as an argument. The `cores` option is set to NULL by default, and can be used to store the number of cores to use as an integer. If `cores` is NULL, facilities from the parallel package will not be used.

Value

The option access functions return their current settings, the assignment functions usually return the previous value of the option.

Note

An extended example is shown in the documentation of `aple.mc`, including treatment of seeding of RNG for multicore/cluster.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
ls(envir=spdep:::.spdepOptions)
if (require(parallel, quietly=TRUE)) {
 nc <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
 nc
# set nc to 1L here
 if (nc > 1L) nc <- 1L
#nc <- ifelse(nc > 2L, 2L, nc)
 coresOpt <- get.coresOption()
 coresOpt
 if (!is.na(nc)) {
  invisible(set.coresOption(nc))
  print(exists("moran.mc"))
  if(.Platform$OS.type == "windows") {
# forking not permitted on Windows - start cluster
   print(get.mcOption())
   cl <- makeCluster(get.coresOption())
   print(clusterEvalQ(cl, exists("moran.mc")))
   set.ClusterOption(cl)
   clusterEvalQ(get.ClusterOption(), library(spdep))
   print(clusterEvalQ(cl, exists("moran.mc")))
   clusterEvalQ(get.ClusterOption(), detach(package:spdep))
   set.ClusterOption(NULL)
   print(clusterEvalQ(cl, exists("moran.mc")))
   stopCluster(cl)
```

```
  } else {
   mcOpt <- get.mcOption()
   print(mcOpt)
   print(mclapply(1:get.coresOption(), function(i) exists("moran.mc"),
    mc.cores=get.coresOption()))
   invisible(set.mcOption(FALSE))
   cl <- makeCluster(nc)
   print(clusterEvalQ(cl, exists("moran.mc")))
   set.ClusterOption(cl)
   clusterEvalQ(get.ClusterOption(), library(spdep))
   print(clusterEvalQ(cl, exists("moran.mc")))
   clusterEvalQ(get.ClusterOption(), detach(package:spdep))
   set.ClusterOption(NULL)
   print(clusterEvalQ(cl, exists("moran.mc")))
   stopCluster(cl)
   invisible(set.mcOption(mcOpt))
  }
  invisible(set.coresOption(coresOpt))
 }
}
```

`set.spChkOption` *Control checking of spatial object IDs*

Description

Provides support for checking the mutual integrity of spatial neighbour weights and spatial data; similar mechanisms are used for passing global verbose and zero.policy options, and for causing functions creating neighbour objects to warn if there are multiple subgraphs.

Usage

```
set.spChkOption(check)
get.spChkOption()
chkIDs(x, listw)
spNamedVec(var, data)
set.VerboseOption(check)
get.VerboseOption()
set.ZeroPolicyOption(check)
get.ZeroPolicyOption()
set.SubgraphOption(check)
get.SubgraphOption()
set.SubgraphCeiling(value)
get.SubgraphCeiling()
set.NoNeighbourOption(check)
get.NoNeighbourOption()
set.listw_is_CsparseMatrix_Option(check)
get.listw_is_CsparseMatrix_Option()
```

Arguments

check	a logical value, TRUE or FALSE
value	an integer value, initialised as 100000L, the sum of the numbers of nodes and edges in the neighbour graph
x	a vector the same length, or a two-dimensional array, or data frame with the same number of rows as the neighbours list in listw
listw	a listw object or nb object inheriting from "nb"
var	a character string or integer value for the column to be selected
data	a two-dimensional array or data frame containing var

Details

Analysis functions will have an spChk argument by default set to NULL, and will call get.spChkOption() to get the global spatial option for whether to check or not — this is initialised to FALSE, and consequently should not break anything. It can be changed to TRUE using set.spChkOption(TRUE), or the spChk argument can be assigned in analysis functions. spNamedVec() is provided to ensure that rownames are passed on to single columns taken from two-dimensional arrays and data frames.

Value

set.spChkOption() returns the old logical value, get.spChkOption() returns the current logical value, and chkIDs() returns a logical value for the test lack of difference. spNamedVec() returns the selected column with the names set to the row names of the object from which it has been extracted.

Note

The motivation for this mechanism is provided by the observation that spatial objects on a map and their attribute data values need to be linked uniquely, to avoid spurious results. The reordering between the legacy Columbus data set used the earlier publications and that available for download from the Spacestat website is just one example of a common problem.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
data(oldcol)
rownames(COL.OLD)
data(columbus, package="spData")
rownames(columbus)
get.spChkOption()
oldChk <- set.spChkOption(TRUE)
get.spChkOption()
chkIDs(COL.OLD, nb2listw(COL.nb))
chkIDs(columbus, nb2listw(col.gal.nb))
chkIDs(columbus, nb2listw(COL.nb))
```

```
tmp <- try(moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb)))
print(tmp)
tmp <- try(moran.test(spNamedVec("CRIME", columbus), nb2listw(col.gal.nb)))
print(tmp)
tmp <- try(moran.test(spNamedVec("CRIME", columbus), nb2listw(COL.nb)))
print(tmp)
set.spChkOption(FALSE)
get.spChkOption()
moran.test(spNamedVec("CRIME", columbus), nb2listw(COL.nb))
tmp <- try(moran.test(spNamedVec("CRIME", columbus), nb2listw(COL.nb),
 spChk=TRUE), silent=TRUE)
print(tmp)
set.spChkOption(oldChk)
get.spChkOption()
```

`skater` *Spatial 'K'luster Analysis by Tree Edge Removal*

Description

This function implements a SKATER procedure for spatial clustering analysis. This procedure essentialy begins with an edges set, a data set and a number of cuts. The output is an object of 'skater' class and is valid for input again.

Usage

```
skater(edges, data, ncuts, crit, vec.crit, method = c("euclidean",
    "maximum", "manhattan", "canberra", "binary", "minkowski",
    "mahalanobis"), p = 2, cov, inverted = FALSE)
```

Arguments

`edges`	A matrix with 2 colums with each row is an edge
`data`	A data.frame with data observed over nodes.
`ncuts`	The number of cuts
`crit`	A scalar or two dimensional vector with criteria for groups. Examples: limits of group size or limits of population size. If scalar, is the minimum criteria for groups.
`vec.crit`	A vector for evaluating criteria.
`method`	Character or function to declare distance method. If `method` is character, method must be "mahalanobis" or "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk". If `method` is one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski", see `dist` for details, because this function as used to compute the distance. If `method="mahalanobis"`, the mahalanobis distance is computed between neighbour areas. If `method` is a `function`, this function is used to compute the distance.

p	The power of the Minkowski distance.
cov	The covariance matrix used to compute the mahalanobis distance.
inverted	logical. If 'TRUE', 'cov' is supposed to contain the inverse of the covariance matrix.

Value

A object of skater class with:

groups	A vector with length equal the number of nodes. Each position identifies the group of node
edges.groups	A list of length equal the number of groups with each element is a set of edges
not.prune	A vector identifying the groups with are not candidates to partition.
candidates	A vector identifying the groups with are candidates to partition.
ssto	The total dissimilarity in each step of edge removal.

Author(s)

Renato M. Assuncao and Elias T. Krainski

References

Assuncao, R.M., Lage J.P., and Reis, E.A. (2002). Analise de conglomerados espaciais via arvore geradora minima. Revista Brasileira de Estatistica, 62, 1-23.

Assuncao, R. M, Neves, M. C., Camara, G. and Freitas, C. da C. (2006). Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees. International Journal of Geographical Information Science Vol. 20, No. 7, August 2006, 797-811

See Also

See Also as mstree

Examples

```
### loading data
(GDAL37 <- as.numeric_version(unname(sf_extSoftVersion()["GDAL"])) >= "3.7.0")
file <- "etc/shapes/bhicv.gpkg.zip"
zipfile <- system.file(file, package="spdep")
if (GDAL37) {
    bh <- st_read(zipfile)
} else {
    td <- tempdir()
    bn <- sub(".zip", "", basename(file), fixed=TRUE)
    target <- unzip(zipfile, files=bn, exdir=td)
    bh <- st_read(target)
}
### data standardized
dpad <- data.frame(scale(as.data.frame(bh)[,5:8]))
```

```
### neighboorhod list
bh.nb <- poly2nb(bh)

### calculating costs
lcosts <- nbcosts(bh.nb, dpad)

### making listw
nb.w <- nb2listw(bh.nb, lcosts, style="B")

### find a minimum spanning tree
mst.bh <- mstree(nb.w,5)

### the mstree plot
par(mar=c(0,0,0,0))
plot(st_geometry(bh), border=gray(.5))
pts <- st_coordinates(st_centroid(bh))
plot(mst.bh, pts, col=2,
     cex.lab=.6, cex.circles=0.035, fg="blue", add=TRUE)
### three groups with no restriction
res1 <- skater(mst.bh[,1:2], dpad, 2)

### groups size
table(res1$groups)

### the skater plot
opar <- par(mar=c(0,0,0,0))
plot(res1, pts, cex.circles=0.035, cex.lab=.7)

### the skater plot, using other colors
plot(res1, pts, cex.circles=0.035, cex.lab=.7,
     groups.colors=heat.colors(length(res1$ed)))

### the Spatial Polygons plot
plot(st_geometry(bh), col=heat.colors(length(res1$edg))[res1$groups])

par(opar)
### EXPERT OPTIONS

### more one partition
res1b <- skater(res1, dpad, 1)

### length groups frequency
table(res1$groups)

table(res1b$groups)

### thee groups with minimum population
res2 <- skater(mst.bh[,1:2], dpad, 2, 200000, bh$Pop)
table(res2$groups)

### thee groups with minimun number of areas
res3 <- skater(mst.bh[,1:2], dpad, 2, 3, rep(1,nrow(bh)))
table(res3$groups)
```

```
### thee groups with minimun and maximun number of areas
res4 <- skater(mst.bh[,1:2], dpad, 2, c(20,50), rep(1,nrow(bh)))
table(res4$groups)

### if I want to get groups with 20 to 40 elements
res5 <- skater(mst.bh[,1:2], dpad, 2,
   c(20,40), rep(1,nrow(bh))) ## DON'T MAKE DIVISIONS
table(res5$groups)

### In this MST don't have groups with this restrictions
### In this case, first I do one division
### with the minimun criteria
res5a <- skater(mst.bh[,1:2], dpad, 1, 20, rep(1,nrow(bh)))
table(res5a$groups)

### and do more one division with the full criteria
res5b <- skater(res5a, dpad, 1, c(20, 40), rep(1,nrow(bh)))
table(res5b$groups)

### and do more one division with the full criteria
res5c <- skater(res5b, dpad, 1, c(20, 40), rep(1,nrow(bh)))
table(res5c$groups)

### It don't have another divison with this criteria
res5d <- skater(res5c, dpad, 1, c(20, 40), rep(1,nrow(bh)))
table(res5d$groups)

## Not run:
data(boston, package="spData")
bh.nb <- boston.soi
dpad <- data.frame(scale(boston.c[,c(7:10)]))
### calculating costs
system.time(lcosts <- nbcosts(bh.nb, dpad))
### making listw
nb.w <- nb2listw(bh.nb, lcosts, style="B")
### find a minimum spanning tree
mst.bh <- mstree(nb.w,5)
### three groups with no restriction
system.time(res1 <- skater(mst.bh[,1:2], dpad, 2))
library(parallel)
nc <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
# set nc to 1L here
if (nc > 1L) nc <- 1L
coresOpt <- get.coresOption()
invisible(set.coresOption(nc))
if(!get.mcOption()) {
# no-op, "snow" parallel calculation not available
  cl <- makeCluster(get.coresOption())
  set.ClusterOption(cl)
}
### calculating costs
system.time(plcosts <- nbcosts(bh.nb, dpad))
```

```
all.equal(lcosts, plcosts, check.attributes=FALSE)
### making listw
pnb.w <- nb2listw(bh.nb, plcosts, style="B")
### find a minimum spanning tree
pmst.bh <- mstree(pnb.w,5)
### three groups with no restriction
system.time(pres1 <- skater(pmst.bh[,1:2], dpad, 2))
if(!get.mcOption()) {
  set.ClusterOption(NULL)
  stopCluster(cl)
}
all.equal(res1, pres1, check.attributes=FALSE)
invisible(set.coresOption(coresOpt))

## End(Not run)
```

`sp.correlogram` *Spatial correlogram*

Description

Spatial correlograms for Moran's I and the autocorrelation coefficient, with print and plot helper functions.

Usage

```
sp.correlogram(neighbours, var, order = 1, method = "corr",
 style = "W", randomisation = TRUE, zero.policy = NULL, spChk=NULL)
## S3 method for class 'spcor'
plot(x, main, ylab, ylim, ...)
## S3 method for class 'spcor'
print(x, p.adj.method="none", ...)
```

Arguments

`neighbours`	an object of class nb
`var`	a numeric vector
`order`	maximum lag order
`method`	"corr" for correlation, "I" for Moran's I, "C" for Geary's C
`style`	`style` can take values W, B, C, and S
`randomisation`	variance of I or C calculated under the assumption of randomisation, if FALSE normality
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors

spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
x	an object from sp.correlogram() of class spcor
p.adj.method	correction method as in p.adjust
main	an overall title for the plot
ylab	a title for the y axis
ylim	the y limits of the plot
...	further arguments passed through

Details

The print function also calculates the standard deviates of Moran's I or Geary's C and a two-sided probability value, optionally using p.adjust to correct by the nymber of lags. The plot function plots a bar from the estimated Moran's I, or Geary's C value to +/- twice the square root of its variance (in previous releases only once, not twice). The table includes the count of included observations in brackets after the lag order. Care needs to be shown when interpreting results for few remaining included observations as lag order increases.

Value

returns a list of class spcor:

res	for "corr" a vector of values; for "I", a matrix of estimates of "I", expectations, and variances
method	"I" or "corr"
cardnos	list of tables of neighbour cardinalities for the lag orders used
var	variable name

Author(s)

Roger Bivand, <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion, pp. 118–122, Martin, R. L., Oeppen, J. E. 1975 The identification of regional forecasting models using space-time correlation functions, *Transactions of the Institute of British Geographers*, 66, 95–118.

See Also

nblag, moran, p.adjust

Examples

```
nc.sids <- st_read(system.file("shapes/sids.gpkg", package="spData")[1], quiet=TRUE)
rn <- as.character(nc.sids$FIPS)
ncCC89_nb <- read.gal(system.file("weights/ncCC89.gal", package="spData")[1],
 region.id=rn)
ft.SID74 <- sqrt(1000)*(sqrt(nc.sids$SID74/nc.sids$BIR74) +
  sqrt((nc.sids$SID74+1)/nc.sids$BIR74))
tr.SIDS74 <- ft.SID74*sqrt(nc.sids$BIR74)
cspc <- sp.correlogram(ncCC89_nb, tr.SIDS74, order=8, method="corr",
 zero.policy=TRUE)
print(cspc)
plot(cspc)
Ispc <- sp.correlogram(ncCC89_nb, tr.SIDS74, order=8, method="I",
 zero.policy=TRUE)
print(Ispc)
print(Ispc, "bonferroni")
plot(Ispc)
Cspc <- sp.correlogram(ncCC89_nb, tr.SIDS74, order=8, method="C",
 zero.policy=TRUE)
print(Cspc)
print(Cspc, "bonferroni")
plot(Cspc)
drop.no.neighs <- !(1:length(ncCC89_nb) %in% which(card(ncCC89_nb) == 0))
sub.ncCC89.nb <- subset(ncCC89_nb, drop.no.neighs)
plot(sp.correlogram(sub.ncCC89.nb, subset(tr.SIDS74,  drop.no.neighs),
 order=8, method="corr"))
```

---

sp.mantel.mc *Mantel-Hubert spatial general cross product statistic*

Description

A permutation test for the spatial general cross product statistic with Moran ($C_{ij} = z_i z_j$), Geary ($C_{ij} = (z_i - z_j)^2$), and Sokal ($C_{ij} = |z_i - z_j|$) criteria, for $z_i = (x_i - \bar{x})/\sigma_x$. `plot.mc.sim` is a helper function to plot the outcomes of the permutation test.

Usage

```
sp.mantel.mc(var, listw, nsim, type = "moran", zero.policy = attr(listw, "zero.policy"),
 alternative = "greater", spChk=NULL, return_boot=FALSE)
## S3 method for class 'mc.sim'
plot(x, xlim, xlab, main, sub, ..., ptype="density")
```

Arguments

`var`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`
`nsim`	number of permutations

type	"moran", "geary" or "sokal" criteria for similarity
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), "two.sided", or "less".
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
return_boot	return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest
x	the object to be plotted
xlim	the range of the x axis
xlab	a title for the x axis
main	an overall title for the plot
sub	a sub title for the plot
ptype	either "density" or "hist"
...	further arguments passed through

Value

A list with class htest and mc.sim containing the following components:

statistic	the value of the observed Geary's C.
parameter	the rank of the observed Geary's C.
alternative	a character string describing the alternative hypothesis.
method	a character string giving the method used.
data.name	a character string giving the name(s) of the data, and the number of simulations.
p.value	the pseudo p-value of the test.
res	nsim simulated values of statistic, final value is observed statistic
estimate	the mean and variance of the simulated distribution.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 22-24, Haining, R. 1990 *Spatial data analysis in the social and environmental sciences*, Cambridge: Cambridge University Press, p. 230–1. The function has been checked against general matrix code posted to the r-help list by Ben Bolker on 1 May 2001; another mantel() function is in the vegan package.

See Also

moran.mc, joincount.mc, geary.mc

Examples

```
data(oldcol)
sim1 <- sp.mantel.mc(COL.OLD$CRIME, nb2listw(COL.nb),
 nsim=99, type="geary", alternative="two.sided")
sim1
plot(sim1)
sp.mantel.mc(COL.OLD$CRIME, nb2listw(COL.nb), nsim=99,
 type="sokal", alternative="two.sided")
sp.mantel.mc(COL.OLD$CRIME, nb2listw(COL.nb), nsim=99,
 type="moran")
```

---

spdep *Return package version number*

Description

The function retreives package version and build information

Usage

```
spdep(build = FALSE)
```

Arguments

build	if TRUE, also returns build information

Value

a character vector with one or two elements

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

---

spdep-defunct *Defunct Functions in Package* **spdep**

Description

These functions are defunct from release 1.2-1. The functions have been moved to the **spatialreg** package.

Usage

```
aple.plot()
localAple()
aple.mc()
aple()
lextrB()
lextrW()
lextrS()
griffith_sone()
subgraph_eigenw()
mom_calc()
mom_calc_int2()
stsls()
## S3 method for class 'stsls'
impacts()
GMerrorsar()
## S3 method for class 'gmsar'
summary()
gstsls()
## S3 method for class 'gmsar'
impacts()
## S3 method for class 'gmsar'
Hausman.test()
lagmess()
ME()
SpatialFiltering()
LR.sarlm()
## S3 method for class 'sarlm'
logLik()
LR1.sarlm()
Wald1.sarlm()
## S3 method for class 'sarlm'
Hausman.test()
as.spam.listw()
as_dgRMatrix_listw()
as_dsTMatrix_listw()
as_dsCMatrix_I()
as_dsCMatrix_IrW()
Jacobian_W()
powerWeights()
## S3 method for class 'lagImpact'
plot()
## S3 method for class 'lagImpact'
print()
## S3 method for class 'lagImpact'
summary()
## S3 method for class 'lagImpact'
HPDinterval()
```

```
intImpacts()
can.be.simmed()
eigenw()
similar.listw()
do_ldet()
jacobianSetup()
cheb_setup()
mcdet_setup()
eigen_setup()
eigen_pre_setup()
spam_setup()
spam_update_setup()
Matrix_setup()
Matrix_J_setup()
LU_setup()
LU_prepermutate_setup()
moments_setup()
SE_classic_setup()
SE_whichMin_setup()
SE_interp_setup()
MCMCsamp()
## S3 method for class 'spautolm'
MCMCsamp()
## S3 method for class 'sarlm'
MCMCsamp()
spautolm()
## S3 method for class 'spautolm'
summary()
spBreg_sac()
## S3 method for class 'MCMC_sar_g'
impacts()
## S3 method for class 'MCMC_sem_g'
impacts()
## S3 method for class 'MCMC_sac_g'
impacts()
spBreg_err()
spBreg_lag()
## S3 method for class 'SLX'
predict()
lmSLX()
## S3 method for class 'SLX'
impacts()
create_WX()
## S3 method for class 'sarlm'
anova()
bptest.sarlm()
errorsarlm()
## S3 method for class 'sarlm'
```

```
impacts()
lagsarlm()
## S3 method for class 'sarlm'
predict()
## S3 method for class 'sarlm.pred'
print()
## S3 method for class 'sarlm.pred'
as.data.frame()
## S3 method for class 'sarlm'
residuals()
## S3 method for class 'sarlm'
deviance()
## S3 method for class 'sarlm'
coef()
## S3 method for class 'sarlm'
vcov()
## S3 method for class 'sarlm'
fitted()
sacsarlm()
## S3 method for class 'sarlm'
summary()
## S3 method for class 'sarlm'
print()
## S3 method for class 'summary.sarlm'
print()
trW()
```

Details

Model-fitting functions and functions supporting model fitting have been moved to the **spatialreg** package.

See Also

Defunct

`spweights.constants` *Provides constants for spatial weights matrices*

Description

The function calculates the constants needed for tests of spatial autocorrelation for general weights matrices represented as `listw` objects. Note: from spdep 0.3-32, the values of S1 and S2 are returned correctly for both underlying symmetric and asymmetric neighbour lists, before 0.3-32, S1 and S2 were wrong for listw objects based on asymmetric neighbour lists, such as k-nearest neighbours (thanks to Luc Anselin for finding the bug).

Usage

```
spweights.constants(listw, zero.policy=attr(listw, "zero.policy"), adjust.n=TRUE)
Szero(listw)
```

Arguments

listw	a listw object from for example nb2listw
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if TRUE ignore zones without neighbours, if FALSE fail when encountered
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

Value

n	number of zones
n1	n - 1
n2	n - 2
n3	n - 3
nn	n * n
S0	global sum of weights
S1	S1 sum of weights
S2	S2 sum of weights

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Haining, R. 1990 Spatial data analysis in the social and environmental sciences, Cambridge University Press, p. 233; Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 19, 21.

See Also

nb2listw

Examples

```
data(oldcol)
B <- spweights.constants(nb2listw(COL.nb, style="B"))
W <- spweights.constants(nb2listw(COL.nb, style="W"))
C <- spweights.constants(nb2listw(COL.nb, style="C"))
S <- spweights.constants(nb2listw(COL.nb, style="S"))
U <- spweights.constants(nb2listw(COL.nb, style="U"))
print(data.frame(rbind(unlist(B), unlist(W), unlist(C), unlist(S), unlist(U)),
  row.names=c("B", "W", "C", "S", "U")))
```

ssw *Compute the sum of dissimilarity*

Description

This function computes the sum of dissimilarity between each observation and the mean (scalar of vector) of the observations.

Usage

```
ssw(data, id, method = c("euclidean", "maximum",
    "manhattan", "canberra", "binary", "minkowski",
    "mahalanobis"), p = 2, cov, inverted = FALSE)
```

Arguments

data	A matrix with observations in the nodes.
id	Node index to compute the cost
method	Character or function to declare distance method. If method is character, method must be "mahalanobis" or "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk". If method is one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowisk", see dist for details, because this function as used to compute the distance. If method="mahalanobis", the mahalanobis distance is computed between neighbour areas. If method is a function, this function is used to compute the distance.
p	The power of the Minkowski distance.
cov	The covariance matrix used to compute the mahalanobis distance.
inverted	logical. If 'TRUE', 'cov' is supposed to contain the inverse of the covariance matrix.

Value

A numeric, the sum of dissimilarity between the observations id of data and the mean (scalar of vector) of this observations.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as nbcost

Examples

```
data(USArrests)
n <- nrow(USArrests)
ssw(USArrests, 1:n)
ssw(USArrests, 1:(n/2))
ssw(USArrests, (n/2+1):n)
ssw(USArrests, 1:(n/2)) + ssw(USArrests, (n/2+1):n)
```

`subset.listw` *Subset a spatial weights list*

Description

The function subsets a spatial weights list, retaining objects for which the subset argument vector is TRUE. At present it will only subset non-general weights lists (that is those created by `nb2listw` with `glist=NULL`).

Usage

```
## S3 method for class 'listw'
subset(x, subset, zero.policy = attr(x, "zero.policy"), ...)
```

Arguments

`x`	an object of class `listw`
`subset`	logical expression
`zero.policy`	default `attr(x, "zero.policy")` as set when `x` was created, if attribute not set, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors - passed through to `nb2listw`
`...`	generic function pass-through

Value

The function returns an object of class `listw` with component `style` the same as the input object, component `neighbours` a list of integer vectors containing neighbour region number ids (compacted to run from 1:number of regions in subset), and component `weights` as the weights computed for `neighbours` using `style`. If no-neighbour observations are created by subsetting and `zero.policy` in the input weights object was FALSE, it will be set to TRUE and a warning issued.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`nb2listw`, `subset.nb`

Examples

```
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
to.be.dropped <- c(31, 34, 36, 39, 42, 46)
pre <- nb2listw(col.gal.nb)
print(pre)
post <- subset(pre, !(1:length(col.gal.nb) %in% to.be.dropped))
print(post)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
nb <- poly2nb(columbus)
lw <- nb2listw(nb, style="W")
attr(lw, "zero.policy")
(lwa <- subset(lw, 1:nrow(columbus) != c(21)))
attr(lwa, "zero.policy")
(lwb <- subset(lw, !(1:nrow(columbus) %in% c(21, 36, 39))))
attr(lwb, "zero.policy")
```

---

subset.nb *Subset a neighbours list*

Description

The function subsets a neighbors list, retaining objects for which the subset argument vector is TRUE.

Usage

```
## S3 method for class 'nb'
subset(x, subset, ...)
```

Arguments

x	an object of class nb
subset	logical expression
...	generic function pass-through

Value

The function returns an object of class nb with a list of integer vectors containing neighbour region number ids (compacted to run from 1:number of regions in subset).

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

nb2listw

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_coordinates(st_centroid(columbus))
plot(col.gal.nb, coords)
to.be.dropped <- c(31, 34, 36, 39, 42, 46)
text(coords[to.be.dropped,1], coords[to.be.dropped,2], labels=to.be.dropped,
  pos=2, offset=0.3)
sub.col.gal.nb <- subset(col.gal.nb,
  !(1:length(col.gal.nb) %in% to.be.dropped))
plot(sub.col.gal.nb, coords[-to.be.dropped,], col="red", add=TRUE)
which(!(attr(col.gal.nb, "region.id") %in%
  attr(sub.col.gal.nb, "region.id")))
```

`summary.nb` *Print and summary function for neighbours and weights lists*

Description

The function prints summary measures for links in a neighbours list. If a matrix of coordinates is given as well, summary descriptive measures for the link lengths are also printed. Print and summary functions are also available for `"listw"` weights list objects, also reporting constants (S0, S1, S2) used in inference for global spatial autocorrelation statistics such as Moran's I, Geary's C, join-count tests and Getis-Ord G.

Usage

```
## S3 method for class 'nb'
summary(object, coords=NULL, longlat = NULL, scale = 1, ...)
## S3 method for class 'nb'
print(x, ...)
## S3 method for class 'listw'
summary(object, coords, longlat, zero.policy = attr(object, "zero.policy"),
 scale = 1, adjust.n=TRUE, ...)
## S3 method for class 'listw'
print(x, zero.policy = attr(x, "zero.policy"), ...)
```

Arguments

`object`	an object of class `nb`
`coords`	matrix of region point coordinates or a SpatialPoints object or an `sfc` points object
`longlat`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if coords is a SpatialPoints object, the value is taken from the object itself
`...`	additional arguments affecting the output produced

x	an object of class nb
zero.policy	default attr(listw, "zero.policy") as set when listw was created, if attribute not set, use global option value; if FALSE stop with error for any empty neighbour sets
scale	passed through to stem() for control of plot length
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations in spweights.constants is adjusted

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

plot.nb

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
col.gal.nb
summary(col.gal.nb, coords)
col.listw <- nb2listw(col.gal.nb, style="W")
col.listw
summary(col.listw)
col_geoms <- st_geometry(columbus)
col_geoms[21] <- st_buffer(col_geoms[21], dist=-0.05)
st_geometry(columbus) <- col_geoms
nb <- poly2nb(columbus)
summary(nb)
try(nb2listw(nb, style="W"))
summary(nb2listw(nb, style="W", zero.policy=TRUE))
summary(nb2listw(nb, style="W", zero.policy=TRUE), adjust.n=FALSE)
```

---

tolerance.nb *Function to construct edges based on a tolerance angle and a maximum distance*

Description

This function creates an object of class nb (defined in the library spdep) containing a connexion diagram. The edges between sites are based on a tolerance angle and a maximum distance. The angle is directional; its direction is always from the bottow to the top of the screen.

Usage

```
tolerance.nb(coords, unit.angle = "degrees", max.dist, tolerance, rot.angle,
 plot.sites=FALSE)
```

Arguments

coords	A matrix or a data frame containing the X and Y coordinates of the study sites.
unit.angle	Character. The measurement units in which angles are defined: either "degrees" (default) or "radians".
max.dist	Numeric. The maximum distance of an edge linking two sites together.
tolerance	Numeric. The tolerance angle in which a site can influence another site. The angle is measured vertically and from bottom to top of the pictures after rotation of the points.
rot.angle	Numeric, optional. An angle at which a set of coordinates should be rotated before creating the connexion diagram. The set of coordinates is rotated counterclockwise. Negative values will produce a clockwise rotation.
plot.sites	Logical (TRUE, FALSE) determining if the site should be plotted in a graphic window. This graph allows one to make sure the points are rotated in a correct direction.

Details

Even though this function creates a connexion diagram based on a tolerance angle going from the bottom to the top of the screen, the resulting object is symmetric, meaning that a site influences another and vice versa. The final object does not represent a directional connexion network.

Value

The function returns an object of class nb with a list of integer vectors corresponding to neighbour region numbers.

Warning

This function was not design to handle a large number of rows in coords. To use this function for a set of coordinates with more than 1500 entries is memory intensive.

Author(s)

F. Guillaume Blanchet

See Also

dnearneigh, cell2nb, graphneigh, tri2nb, knn2nb

Examples

```
set.seed(1)
ex.data<-cbind(runif(50),rexp(50))

### Construct object of class nb with a tolerance angle of 30 degrees
### and a maximum distance of 2 m.
nb.ex<-tolerance.nb(ex.data, unit.angle = "degrees", max.dist=1,
 tolerance = 30)

### Construct object of class nb with a tolerance angle of 30 degrees
### and a maximum distance of 2 m. The coordinates are rotated at an angle
### of 45 degrees counterclockwise.
nb.ex2<-tolerance.nb(ex.data, unit.angle = "degrees", max.dist=1,
 tolerance = 30, rot.angle = 45)

### Construct object of class nb with a tolerance angle of pi/8 radians
### and a maximum distance of 1.5 m. The coordinates are rotated at
### an angle of pi/4 radians clockwise.
nb.ex3<-tolerance.nb(ex.data, unit.angle = "radians", max.dist=1.5,
 tolerance = pi/8, rot.angle = -pi*2/3)

par(mfrow=c(1,3))
plot(nb.ex,ex.data,asp=1)
plot(nb.ex2,ex.data,asp=1)
plot(nb.ex3,ex.data,asp=1)
```

---

tri2nb *Neighbours list from tri object*

Description

The function uses the `deldir` package to convert a matrix of two-dimensional coordinates into a neighbours list of class `nb` with a list of integer vectors containing neighbour region number ids.

Usage

```
tri2nb(coords, row.names = NULL)
```

Arguments

`coords`	matrix, data.frame or tibble of point coordinates with two columns, a SpatialPoints object or an `sfc` points object
`row.names`	character vector of region ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))`

Details

If coordinates are duplicated, this function cannot be used. If the coordinates are from a grid, then they need to be ordered such that the first three are not collinear, so that the first triangle can be constructed. This can be achieved by randomising the order of the coordinates (possibly several times), and then re-ordering the order of the data to match the new order of the neighbour list - if this fix is used, remember to re-order the row.names argument as well as the coordinates! Please also note that triangulation of grid points will give arbitrary diagonal neighbours, which may not be a sensible outcome, and `dnearneigh()` may serve better where `tri2nb()` cannot be used.

Value

The function returns an object of class `nb` with a list of integer vectors containing neighbour region number ids.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knn2nb, dnearneigh, cell2nb

Examples

```
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
ind <- row.names(columbus)
suppressPackageStartupMessages(require(deldir))
col.tri.nb <- tri2nb(coords, row.names=ind)
plot(st_geometry(columbus), border="grey")
plot(col.tri.nb, coords, add=TRUE)
title(main="Raw triangulation links", cex.main=0.6)
x <- seq(0,1,0.1)
y <- seq(0,2,0.2)
xy <- expand.grid(x, y)
try(xy.nb <- tri2nb(xy))
seed <- 1234
xid <- sample(1:nrow(xy))
xy.nb <- tri2nb(xy[xid,])
plot(xy.nb, xy[xid,])
# example of reading points with readr::read_csv() yielding a tibble
load(system.file("etc/misc/coords.rda", package="spdep"))
class(coords)
nb <- tri2nb(coords)
plot(nb, coords)
```

write.nb.gal *Write a neighbours list as a GAL lattice file*

Description

Write a neighbours list as a GAL lattice file, may also use newer GeoDa header format

Usage

```
write.nb.gal(nb, file, oldstyle=TRUE, shpfile=NULL, ind=NULL)
```

Arguments

nb	an object of class nb with a list of integer vectors containing neighbour region number ids.
file	name of file with GAL lattice data
oldstyle	if TRUE, first line of file contains only number of spatial units, if FALSE, uses newer GeoDa style
shpfile	Shapefile name taken from GAL file for this dataset
ind	region id indicator variable name

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

read.gal

Examples

```
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
GALfile <- tempfile("GAL")
write.nb.gal(col.gal.nb, GALfile)
col.queen <- read.gal(GALfile)
summary(diffnb(col.queen, col.gal.nb))
```

Index