

# Package ‘spdep’

June 29, 2020

**Version** 1.1-5

**Date** 2020-06-25

**Title** Spatial Dependence: Weighting Schemes, Statistics

**Encoding** UTF-8

**Depends** R (>= 3.3.0), methods, sp (>= 1.0), spData (>= 0.2.6.0), sf

**Imports** deldir, boot (>= 1.3-1), graphics, grDevices, utils, stats, Matrix, LearnBayes, MASS, coda, expm, gmodels, nlme

**Suggests** spatialreg (>= 1.1-1), parallel, RANN, rgeos, RColorBrewer, lattice, xtable, maptools (>= 0.5-4), foreign, igraph, RSpectra, knitr, rgdal, classInt, tmap, spam, ggplot2, markdown

**URL** <https://github.com/r-spatial/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' ('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  
 the measures is described in 'Bivand' and 'Wong' (2018)  
 <doi:10.1007/s11749-018-0599-x>.

'spdep' >= 1.1-1 corresponds to 'spatialreg' >= 1.1-1, in which the model  
 fitting functions are deprecated and pass through to 'spatialreg', but  
 will mask those in 'spatialreg'. From versions 1.2-1, the functions will  
 be made defunct in 'spdep'.

For now 'spatialreg' only has functions from 'spdep', where they are shown  
 as deprecated. 'spatialreg' only loads the namespace of 'spdep'; if you  
 attach 'spdep', the same functions in the other package will be masked.  
 Some feed through adequately, others do not.

**License** GPL (>= 2)

**VignetteBuilder** knitr

**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],  
 Olaf Berke [ctb],  
 Andrew Bernat [ctb],  
 Guillaume Blanchet [ctb],  
 Eric Blankmeyer [ctb],  
 Marília Carvalho [ctb],  
 Bjarke Christensen [ctb],  
 Yongwan Chun [ctb],  
 Carsten Dormann [ctb],  
 Stéphane Dray [ctb],  
 Virgilio Gómez-Rubio [ctb],  
 Martin Gubri [ctb],  
 Rein Halbersma [ctb],  
 Elias Krainski [ctb],  
 Pierre Legendre [ctb],  
 Nicholas Lewin-Koh [ctb],  
 Angela Li [ctb],  
 Hongfei Li [ctb],  
 Jielai Ma [ctb],  
 Abhirup Mallik [ctb, trl],  
 Giovanni Millo [ctb],  
 Werner Mueller [ctb],  
 Hisaji Ono [ctb],  
 Pedro Peres-Neto [ctb],  
 Gianfranco Piras [ctb],  
 Markus Reeder [ctb],  
 Michael Tiefelsdorf [ctb],  
 René Westerholt [ctb],

Danlin Yu [ctb]

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

**Repository** CRAN

**Date/Publication** 2020-06-29 10:10:03 UTC

## R topics documented:

aggregate.nb . . . . .	5
airdist . . . . .	6
aple . . . . .	7
aple.mc . . . . .	8
aple.plot . . . . .	10
autocov_dist . . . . .	11
bhcv . . . . .	13
card . . . . .	14
cell2nb . . . . .	15
choynowski . . . . .	16
columbus . . . . .	18
diffnb . . . . .	18
dnearneigh . . . . .	19
droplinks . . . . .	21
EBest . . . . .	22
EBImoran.mc . . . . .	23
EBlocal . . . . .	26
edit.nb . . . . .	27
eire . . . . .	28
geary . . . . .	28
geary.mc . . . . .	30
geary.test . . . . .	31
globalG.test . . . . .	33
Graph Components . . . . .	35
graphneigh . . . . .	36
grid2nb . . . . .	39
include.self . . . . .	41
invIrM . . . . .	42
is.symmetric.nb . . . . .	44
joincount.mc . . . . .	46
joincount.multi . . . . .	47
joincount.test . . . . .	49
knearneigh . . . . .	51
knn2nb . . . . .	53
lag.listw . . . . .	54
lee . . . . .	55
lee.mc . . . . .	56
lee.test . . . . .	58
listw2sn . . . . .	60
lm.LMtests . . . . .	61

lm.morantest	63
lm.morantest.exact	65
lm.morantest.sad	67
localG	69
localmoran	71
localmoran.exact	73
localmoran.sad	76
LOSH	79
LOSH.cs	81
LOSH.mc	82
mat2listw	84
moran	85
moran.mc	86
moran.plot	88
moran.test	90
mstree	92
nb.set.operations	94
nb2blocknb	95
nb2INLA	96
nb2lines	97
nb2listw	99
nb2mat	101
nb2WB	102
nbcosts	103
nbdists	104
nblag	105
oldcol	106
p.adjustSP	108
plot.mst	109
plot.nb	110
plot.skater	111
poly2nb	112
probmap	113
prunecost	115
prunemst	116
read.gal	117
read.gwt2nb	118
Rotation	120
set.mcOption	121
set.spChkOption	123
skater	125
sp.correlogram	129
sp.mantel.mc	131
spdep	133
spdep-deprecated	133
spweights.constants	164
ssw	165
subset.listw	166

*aggregate.nb* 5

subset.nb . . . . .	167
summary.nb . . . . .	168
tolerance.nb . . . . .	169
tri2nb . . . . .	171
write.nb.gal . . . . .	172

**Index** 174

---

*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

<code>x</code>	an nb neighbour object
<code>IDs</code>	a character vector of IDs grouping the members of the neighbour object
<code>remove.self</code>	default TRUE: remove self-neighbours resulting from aggregation
<code>...</code>	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](#)

---

aple *Approximate profile-likelihood estimator (APLE)*

---

### Description

The Approximate profile-likelihood estimator (APLE) of the simultaneous autoregressive model's spatial dependence parameter was introduced in Li et al. (2007). It employs a correction term using the eigenvalues of the spatial weights matrix, and consequently should not be used for large numbers of observations. It also requires that the variable has a mean of zero, and it is assumed that it has been detrended. The spatial weights object is assumed to be row-standardised, that is using default style="W" in nb2listw.

### Usage

```
aple(x, listw, override_similarity_check=FALSE, useTrace=TRUE)
```

### Arguments

x	a zero-mean detrended continuous variable
listw	a listw object from for example nb2listw
override\_similarity\_check	default FALSE, if TRUE - typically for row-standardised weights with asymmetric underlying general weights - similarity is not checked
useTrace	default TRUE, use trace of sparse matrix $W \%* \% W$ (Li et al. (2010)), if FALSE, use crossproduct of eigenvalues of $W$ as in Li et al. (2007)

### Details

This implementation has been checked with Hongfei Li's own implementation using her data; her help was very valuable.

### Value

A scalar APLE value.

### Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

### References

Li, H, Calder, C. A. and Cressie N. A. C. (2007) Beyond Moran's I: testing for spatial dependence based on the spatial autoregressive model. *Geographical Analysis* 39, 357-375; Li, H, Calder, C. A. and Cressie N. A. C. (2012) One-step estimation of spatial dependence parameters: Properties and extensions of the APLE statistic, *Journal of Multivariate Analysis* 105, 68-84.

**See Also**

[nb2listw](#), [a`ple.mc`](#), [a`ple.plot`](#)

**Examples**

```
wheat <- st_read(system.file("shapes/wheat.shp", package="spData")[1], quiet=TRUE)
nbr1 <- poly2nb(wheat, queen=FALSE)
nbrl <- nblag(nbr1, 2)
nbr12 <- nblag_cumul(nbrl)
cms0 <- with(as.data.frame(wheat), tapply(yield, c, median))
cms1 <- c(model.matrix(~ factor(c) -1, data=wheat) %*% cms0)
wheat$yield_detrend <- wheat$yield - cms1
isTRUE(all.equal(c(with(as.data.frame(wheat),
  tapply(yield_detrend, c, median))), rep(0.0, 25),
  check.attributes=FALSE))
moran.test(wheat$yield_detrend, nb2listw(nbr12, style="W"))
aple(as.vector(scale(wheat$yield_detrend, scale=FALSE)), nb2listw(nbr12, style="W"))
## Not run:
errorsarlm(yield_detrend ~ 1, wheat, nb2listw(nbr12, style="W"))

## End(Not run)
```

---

a`ple.mc`

*Approximate profile-likelihood estimator (APLE) permutation test*

---

**Description**

A permutation bootstrap test for the approximate profile-likelihood estimator (APLE).

**Usage**

```
aple.mc(x, listw, nsim, override_similarity_check=FALSE, useTrace=TRUE)
```

**Arguments**

<code>x</code>	a zero-mean detrended continuous variable
<code>listw</code>	a <code>listw</code> object from for example <code>nb2listw</code>
<code>nsim</code>	number of simulations
<code>override\_similarity\_check</code>	default FALSE, if TRUE - typically for row-standardised weights with asymmetric underlying general weights - similarity is not checked
<code>useTrace</code>	default TRUE, use trace of sparse matrix $W \%* \% W$ (Li et al. (2010)), if FALSE, use crossproduct of eigenvalues of $W$ as in Li et al. (2007)

**Value**

A boot object as returned by the boot function.



**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**References**

Li, H, Calder, C. A. and Cressie N. A. C. (2007) Beyond Moran's I: testing for spatial dependence based on the spatial autoregressive model. *Geographical Analysis* 39, 357-375; Li, H, Calder, C. A. and Cressie N. A. C. (2012) One-step estimation of spatial dependence parameters: Properties and extensions of the APLE statistic, *Journal of Multivariate Analysis* 105, 68-84.

**See Also**

[apple](#), [boot](#)

**Examples**

```
## Not run:
wheat <- st_read(system.file("shapes/wheat.shp", package="spData")[1], quiet=TRUE)
nbr1 <- poly2nb(wheat, queen=FALSE)
nbr1 <- nblag(nbr1, 2)
nbr12 <- nblag_cumul(nbr1)
wheat_g <- wheat
st_geometry(wheat_g) <- NULL
cms0 <- with(wheat_g, tapply(yield, c, median))
cms1 <- c(model.matrix(~ factor(c) -1, data=wheat) %**% cms0)
wheat$yield_detrend <- wheat$yield - cms1
oldRNG <- RNGkind()
RNGkind("L'Ecuyer-CMRG")
set.seed(1L)
boot_out_ser <- apple.mc(as.vector(scale(wheat$yield_detrend, scale=FALSE)),
  nb2listw(nbr12, style="W"), nsim=500)
plot(boot_out_ser)
boot_out_ser
library(parallel)
oldCores <- set.coresOption(NULL)
nc <- detectCores(logical=FALSE)
# set nc to 1L here
if (nc > 1L) nc <- 1L
invisible(set.coresOption(nc))
set.seed(1L)
if (!get.mcOption()) {
  cl <- makeCluster(nc)
  set.ClusterOption(cl)
} else{
  mc.reset.stream()
}
boot_out_par <- apple.mc(as.vector(scale(wheat$yield_detrend, scale=FALSE)),
  nb2listw(nbr12, style="W"), nsim=500)
if (!get.mcOption()) {
  set.ClusterOption(NULL)
  stopCluster(cl)
}
```

```

boot_out_par
invisible(set.coresOption(oldCores))
RNGkind(oldRNG[1], oldRNG[2])

## End(Not run)

```

---

apple.plot

*Approximate profile-likelihood estimator (APLE) scatterplot*


---

### Description

A scatterplot decomposition of the approximate profile-likelihood estimator, and a local APLE based on the list of vectors returned by the scatterplot function.

### Usage

```

apple.plot(x, listw, override_similarity_check=FALSE, useTrace=TRUE, do.plot=TRUE, ...)
localAple(x, listw, override_similarity_check=FALSE, useTrace=TRUE)

```

### Arguments

x	a zero-mean detrended continuous variable
listw	a listw object from for example nb2listw
override\_similarity\_check	default FALSE, if TRUE - typically for row-standardised weights with asymmetric underlying general weights - similarity is not checked
useTrace	default TRUE, use trace of sparse matrix $W \%* \% W$ (Li et al. (2010)), if FALSE, use crossproduct of eigenvalues of $W$ as in Li et al. (2007)
do.plot	default TRUE: should a scatterplot be drawn
...	other arguments to be passed to plot

### Details

The function solves a secondary eigenproblem of size  $n$  internally, so constructing the values for the scatterplot is quite compute and memory intensive, and is not suitable for very large  $n$ .

### Value

apple.plot returns list with components:

X	A vector as described in Li et al. (2007), p. 366.
Y	A vector as described in Li et al. (2007), p. 367.

localAple returns a vector of local APLE values.

### Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

## References

Li, H, Calder, C. A. and Cressie N. A. C. (2007) Beyond Moran's I: testing for spatial dependence based on the spatial autoregressive model. *Geographical Analysis* 39, pp. 357-375; Li, H, Calder, C. A. and Cressie N. A. C. (2012) One-step estimation of spatial dependence parameters: Properties and extensions of the APLE statistic, *Journal of Multivariate Analysis* 105, 68-84.

## See Also

[aple](#)

## Examples

```
## Not run:
wheat <- st_read(system.file("shapes/wheat.shp", package="spData")[1], quiet=TRUE)
nbr1 <- poly2nb(wheat, queen=FALSE)
nbr1 <- nblag(nbr1, 2)
nbr12 <- nblag_cumul(nbr1)
cms0 <- with(as.data.frame(wheat), tapply(yield, c, median))
cms1 <- c(model.matrix(~ factor(c) -1, data=wheat) %*% cms0)
wheat$yield_detrend <- wheat$yield - cms1
plt_out <- aple.plot(as.vector(scale(wheat$yield_detrend, scale=FALSE)),
  nb2listw(nbr12, style="W"), cex=0.6)
lm_obj <- lm(Y ~ X, plt_out)
abline(lm_obj)
abline(v=0, h=0, lty=2)
zz <- summary(influence.measures(lm_obj))
infl <- as.integer(rownames(zz))
points(plt_out$X[infl], plt_out$Y[infl], pch=3, cex=0.6, col="red")
crossprod(plt_out$Y, plt_out$X)/crossprod(plt_out$X)
wheat$localAple <- localAple(as.vector(scale(wheat$yield_detrend, scale=FALSE)),
  nb2listw(nbr12, style="W"))
mean(wheat$localAple)
hist(wheat$localAple)
opar <- par(no.readonly=TRUE)
plot(wheat[, "localAple"], reset=FALSE)
text(st_coordinates(st_centroid(st_geometry(wheat)))[infl,], labels=rep("*", length(infl)))
par(opar)

## End(Not run)
```

---

autocov\_dist

*Distance-weighted autocovariate*

---

## Description

Calculates the autocovariate to be used in autonormal, autopoission 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., Muggleston 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.shp", 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")
acinva2a <- 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$CRIME, pch=16, col="red")
points(acinva2a ~ columbus$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")
acinva2b <- lag(lw, columbus$CRIME)
all.equal(acinva2b, acinva2a)
xy <- SpatialPoints(xy)
acinva <- autocov_dist(columbus$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 in the Atlas of condition indices published by the Joao Pinheiro Foundation and UNDP.

**Format**

A shape polygon object with seven variables:

**id** The identifier

**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**

```
bh <- st_read(system.file("etc/shapes/bhcv.shp",
  package="spdep")[1])
```

---

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.

## 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)
mrc2vi(rowcol, nrow, ncol)
rookcell(rowcol, nrow, ncol, torus=FALSE, rmin=1, cmin=1)
queencell(rowcol, nrow, ncol, torus=FALSE, rmin=1, cmin=1)
vi2mrc(i, nrow, ncol)
```

## Arguments

nrow	number of rows in the grid
ncol	number of columns in the grid
type	rook or queen
torus	map grid onto torus
legacy	default FALSE, nrow/ncol reversed, if TRUE wrong col/row directions (see <a href="https://github.com/r-spatial/spdep/issues/20">https://github.com/r-spatial/spdep/issues/20</a> )
rowcol	matrix with two columns of row, column indices
i	vector of vector indices corresponding to rowcol
rmin	lowest row index
cmin	lowest column index

**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)
# 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
t.nb <- cell2nb(nrow=50, ncol=10, type='rook', legacy=TRUE)
isTRUE(all.equal(nb_rook_cont, t.nb, check.attributes=FALSE))
## [1] FALSE
t.nb <- cell2nb(nrow=50, ncol=10, type='rook')
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**

<code>n</code>	a numeric vector of counts of cases
<code>x</code>	a numeric vector of populations at risk
<code>row.names</code>	row names passed through to output data frame
<code>tol</code>	accumulate values for observed counts $\geq$ expected until value less than tol
<code>legacy</code>	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:

<code>pmap</code>	Poisson probability map values: probability of getting a more “extreme” count than actually observed, one-tailed with less than expected and more than expected folded together
<code>type</code>	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.shp", package="spData")[1], quiet=TRUE)
auckland.nb <- poly2nb(auckland)
res <- choynowski(auckland$M77_85, 9*auckland$Und5_81)
res1 <- choynowski(auckland$M77_85, 9*auckland$Und5_81, legacy=TRUE)
all.equal(res, res1)
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.shp", 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.shp", 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 between lower (greater than) and upper (less than or equal to) bounds, or with `longlat = TRUE`, by Great Circle distance in kilometers.

**Usage**

```
dnearneigh(x, d1, d2, row.names = NULL, longlat = NULL, bounds=c("GT", "LE"))
```

**Arguments**

x	matrix of point coordinates or a <code>SpatialPoints</code> object or an <code>sf</code> or <code>sfc</code> points object
d1	lower distance bound
d2	upper distance bound
row.names	character vector of region ids to be added to the neighbours list as attribute <code>region.id</code> , default <code>seq(1, nrow(x))</code>
longlat	<code>TRUE</code> if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if <code>x</code> is a <code>SpatialPoints</code> object, the value is taken from the object itself, and overrides this argument if not <code>NULL</code>
bounds	character vector of length 2, default <code>c("GT", "LE")</code> , the first element may also be <code>"GE"</code> , the second <code>"LT"</code>

**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.shp", 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)
```

---

droplinks

*Drop links in a neighbours list*

---

### Description

Drops links to and from or just to a region from a neighbours list. The example corresponds to Fingleton's Table 1, p. 6, for lattices 5 to 19.

### Usage

```
droplinks(nb, drop, 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

### 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 Gattrell 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.shp", 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
```

**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 = NULL,
            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 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), 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:

$$p_i = n_i / x_i$$

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

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

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

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



**Value**

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

<code>statistic</code>	the value of the observed Moran's I.
<code>parameter</code>	the rank of the observed Moran's I.
<code>p.value</code>	the pseudo p-value of the test.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the method used.
<code>data.name</code>	a character string giving the name(s) of the data, and the number of simulations.
<code>res</code>	<code>nsim</code> simulated values of statistic, final value is observed statistic
<code>z</code>	a numerical vector of Empirical Bayes indices as <code>z</code> 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 <https://doi.org/10.1007/s11749-018-0599-x>

**See Also**

[moran](#), [moran.mc](#), [EBest](#)

**Examples**

```
nc.sids <- st_read(system.file("shapes/sids.shp", 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$BIR74,
  nb2listw(ncCC89_nb, style="B", zero.policy=TRUE), nsim=999, zero.policy=TRUE)
sids.p <- nc.sids$SID74 / nc.sids$BIR74
moran.mc(sids.p, nb2listw(ncCC89_nb, style="B", zero.policy=TRUE),
  nsim=999, zero.policy=TRUE)
```

---

EBlocal	<i>Local Empirical Bayes estimator</i>
---------	--

---

### 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
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 <code>get.spChkOption()</code>
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: <a href="http://agec221.agecon.uiuc.edu/pipermail/openspace/2003-June/thread.html">http://agec221.agecon.uiuc.edu/pipermail/openspace/2003-June/thread.html</a> )

### 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.

### 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:

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.shp", 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**

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

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

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

**Usage**

```
geary(x, listw, n, n1, S0, zero.policy=NULL)
```

**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 NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

**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=NULL, alternative="greater",
         spChk=NULL, adjust.n=TRUE, return_boot=FALSE)
```

**Arguments**

<code>x</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>nsim</code>	number of permutations
<code>zero.policy</code>	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "greater" (default), or "less"; this reversal corresponds to that on <a href="#">geary.test</a> 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).
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>
<code>adjust.n</code>	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
<code>return_boot</code>	return an object of class <code>boot</code> from the equivalent permutation bootstrap rather than an object of class <code>htest</code>

**Value**

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

<code>statistic</code>	the value of the observed Geary's C.
<code>parameter</code>	the rank of the observed Geary's C.
<code>p.value</code>	the pseudo p-value of the test.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the method used.
<code>data.name</code>	a character string giving the name(s) of the data, and the number of simulations.
<code>res</code>	<code>nsim</code> 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)
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)
```

---

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=NULL,
  alternative="greater", spChk=NULL, adjust.n=TRUE)
```

**Arguments**

<code>x</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>randomisation</code>	variance of I calculated under the assumption of randomisation, if FALSE normality
<code>zero.policy</code>	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>
<code>adjust.n</code>	default TRUE, if FALSE the number of observations is not adjusted for non-neighbour observations, if TRUE, the number of observations is adjusted

**Value**

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

<code>statistic</code>	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.
<code>p.value</code>	the p-value of the test.
<code>estimate</code>	the value of the observed Geary's C, its expectation and variance under the method assumption.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the assumption used for calculating the standard deviate.
<code>data.name</code>	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 <https://doi.org/10.1007/s11749-018-0599-x>

## 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)
```

---

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=NULL, alternative="greater",
  spChk=NULL, adjust.n=TRUE, B1correct=TRUE, adjust.x=TRUE, Arc_all_x=FALSE)
```

**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 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()
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

**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 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 <https://doi.org/10.1007/s11749-018-0599-x>

**See Also**

[localG](#)

**Examples**

```

nc.sids <- st_read(system.file("shapes/sids.shp", 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))))

```

**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`

**Value**

A list of:

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

**Author(s)**

Nicholas Lewin-Koh <nikko@hailmail.net>

**See Also**[plot.nb](#)**Examples**

```

columbus <- st_read(system.file("shapes/columbus.shp", 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)) run <- TRUE
if (run) {
  B <- as(nb2listw(col2, style="B", zero.policy=TRUE), "CsparseMatrix")
  g1 <- graph.adjacency(B, mode="undirected")
  c1 <- clusters(g1)
  print(c1$no == res$nc)
}
if (run) {
  print(all.equal(c1$membership, res$comp.id))
}
if (run) {
  print(all.equal(c1$csizes, c(table(res$comp.id)), check.attributes=FALSE))
}
if (run) {
  W <- as(nb2listw(col2, style="W", zero.policy=TRUE), "CsparseMatrix")
  g1W <- graph.adjacency(W, mode="directed", weighted="W")
  c1W <- clusters(g1W)
  print(all.equal(c1W$membership, res$comp.id, check.attributes=FALSE))
}
if (run) {
  B1 <- get.adjacency(g1)
  print(all.equal(B, B1))
}

```

**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 <a href="#">geos_unary</a>
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 symmetric (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 neighbors 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 **RANN** and **rgeos**; from 2019-01-21, **rgeos** replaced by equivalent functions in **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

<code>np</code>	number of input points
<code>from</code>	array of origin ids
<code>to</code>	array of destination ids
<code>nedges</code>	number of edges in graph
<code>x</code>	input x coordinates
<code>y</code>	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](mailto: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.shp", 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(RANN, 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)
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)

```

**Description**

The function builds a neighbours list for a grid topology. It works for a k-dimensional grid topology,  $k \geq 1$ .

**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 supplying the dimensions in the d argument.
d	A scalar (for one dimensional grid) or a length k vector specifying the number of grid cells in each direction of the k dimensions.
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-dimensional 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)

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))
```

---

include.self	<i>Include self in neighbours list</i>
--------------	--

---

## Description

The function includes the region itself in its own list of neighbours, and sets attribute "self.included" to TRUE.

## Usage

```
include.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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
summary(col.gal.nb, coords)
summary(include.self(col.gal.nb), coords)
```

---

 invIrM

 Compute SAR generating operator
 

---

### Description

Computes the matrix used for generating simultaneous autoregressive random variables, for a given value of rho, a neighbours list object or a matrix, a chosen coding scheme style, and optionally a list of general weights corresponding to neighbours.

### Usage

```
invIrM(neighbours, rho, glist=NULL, style="W", method="solve",
       feasible=NULL)
invIrW(x, rho, method="solve", feasible=NULL)
```

### Arguments

neighbours	an object of class nb
rho	autoregressive parameter
glist	list of general weights corresponding to neighbours
style	style can take values W, B, C, and S
method	default solve, can also take value chol
feasible	if NULL, the given value of rho is checked to see if it lies within its feasible range, if TRUE, the test is not conducted
x	either a listw object from for example nb2listw, or a square spatial weights matrix, optionally a sparse matrix

### Details

The `invIrW` function generates the full weights matrix  $V$ , checks that rho lies in its feasible range between  $1/\min(\text{eigen}(V))$  and  $1/\max(\text{eigen}(V))$ , and returns the  $n \times n$  inverted matrix

$$(I - \rho V)^{-1}$$

. With `method="chol"` (only for a listw object), Cholesky decomposition is used, thanks to contributed code by Markus Reder and Werner Mueller.

Note that, in some situations in simulation, it may matter that the random vector from `rnorm` or similar will not be exactly  $N(0, 1)$ , and it will also contain random amounts of spatial autocorrelation itself, which will mix with the spatial autocorrelation injected by the process operator

$$(I - \rho V)^{-1}$$

. In addition, it will not follow the stipulated distribution exactly either, so that several steps may be needed to scale the random vector, to remove artefacts coming from its deviance from distributional parameters, and to remove random spatial autocorrelation - see the examples below. Thanks to Rune Østergaard Pedersen for bring up this question.

The `powerWeights` function uses power series summation to cumulate the product

$$(I - \rho V)^{-1} \% * \% X$$

from

$$(I + \rho V + (\rho V)^2 + \dots) \% * \% X$$

, which can be done by storing only sparse  $V$  and several matrices of the same dimensions as  $X$ . This makes it possible to handle larger spatial weights matrices, but is sensitive to the power weights order and the tolerance arguments when the spatial coefficient is close to its bounds, leading to incorrect estimates of the implied inverse matrix.

### Value

An  $n \times n$  matrix with a "call" attribute; the `powerWeights` function returns a matrix of the same dimensions as  $X$  which has been multiplied by the power series equivalent of the dense matrix

$$(I - \rho V)^{-1}$$

.

### Note

Before version 0.6-10, `powerWeights` only worked correctly for positive  $\rho$ , with differences from true values increasing as  $\rho$  approached -1, and exploding between -1 and the true negative bound.

### 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; Tiefelsdorf, M. 2000 Modelling spatial processes, *Lecture notes in earth sciences*, Springer, p. 76; Haining, R. 1990 *Spatial data analysis in the social and environmental sciences*, Cambridge University Press, p. 117; Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion, p. 152; Reder, M. and Mueller, W. (2007) An Improvement of the `invIrM` Routine of the Geostatistical R-package `spdep` by Cholesky Inversion, *Statistical Projects*, LV No: 238.205, SS 2006, Department of Applied Statistics, Johannes Kepler University, Linz

### See Also

[nb2listw](#)

### Examples

```
nb7rt <- cell2nb(7, 7, torus=TRUE)
lw <- nb2listw(nb7rt, style="W")
set.seed(1)
x <- matrix(sample(rnorm(500*length(nb7rt))), nrow=length(nb7rt))
if (requireNamespace("spatialreg", quietly=TRUE)) {
# Only needed in some simulation settings where the input and
```

```

# output distributions must agree in all but autocorrelation
if (FALSE) {
e <- spatialreg::eigenw(lw)
x <- apply(x, 2, scale)
st <- apply(x, 2, function(x) shapiro.test(x)$p.value)
x <- x[, (st > 0.2 & st < 0.8)]
x <- apply(x, 2, function(v) spatialreg::residuals.spautolm(
  spatialreg::spautolm(v ~ 1, listw=lw, method="eigen",
    control=list(pre_eig=e, fdHess=FALSE))))
x <- apply(x, 2, scale)
}
res0 <- apply(invIrM(nb7rt, rho=0.0, method="chol",
  feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res2 <- apply(invIrM(nb7rt, rho=0.2, method="chol",
  feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res4 <- apply(invIrM(nb7rt, rho=0.4, method="chol",
  feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res6 <- apply(invIrM(nb7rt, rho=0.6, method="chol",
  feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res8 <- apply(invIrM(nb7rt, rho=0.8, method="chol",
  feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res9 <- apply(invIrM(nb7rt, rho=0.9, method="chol",
  feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
plot(density(res9), col="red", xlim=c(-0.01, max(density(res9)$x)),
  ylim=range(density(res0)$y),
  xlab="estimated variance of the mean",
  main=expression(paste("Effects of spatial autocorrelation for different ",
    rho, " values")))
lines(density(res0), col="black")
lines(density(res2), col="brown")
lines(density(res4), col="green")
lines(density(res6), col="orange")
lines(density(res8), col="pink")
legend(c(-0.02, 0.01), c(7, 25),
  legend=c("0.0", "0.2", "0.4", "0.6", "0.8", "0.9"),
  col=c("black", "brown", "green", "orange", "pink", "red"), lty=1, bty="n")
}
## Not run:
x <- matrix(rnorm(length(nb7rt)), ncol=1)
system.time(e <- invIrM(nb7rt, rho=0.9, method="chol", feasible=TRUE) %*% x)
system.time(e <- invIrM(nb7rt, rho=0.9, method="chol", feasible=NULL) %*% x)
system.time(e <- invIrM(nb7rt, rho=0.9, method="solve", feasible=TRUE) %*% x)
system.time(e <- invIrM(nb7rt, rho=0.9, method="solve", feasible=NULL) %*% x)

## End(Not run)

```

## 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

<code>nb</code>	an object of class <code>nb</code> with a list of integer vectors containing neighbour region number ids.
<code>verbose</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> prints non-matching pairs
<code>force</code>	do not respect a neighbours list <code>sym</code> attribute and test anyway
<code>glist</code>	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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
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=FALSE, alternative="greater",
  spChk=NULL)
```

## Arguments

<code>fx</code>	a factor of the same length as the neighbours and weights objects in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>nsim</code>	number of permutations
<code>zero.policy</code>	if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "greater" (default), or "less".
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>

## Value

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

<code>statistic</code>	the value of the observed statistic.
<code>parameter</code>	the rank of the observed statistic.
<code>method</code>	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)
joincount.test(HICRIME, nb2listw(COL.nb, style="B"))
```

---

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 = FALSE,
  spChk = NULL, adjust.n=TRUE)
## S3 method for class 'jcmulti'
print(x, ...)
```

**Arguments**

<code>fx</code>	a factor of the same length as the neighbours and weights objects in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>zero.policy</code>	if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
<code>adjust.n</code>	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 <code>spdep</code> 0.3-28 the adjustment was inconsistent - thanks to Tomoki NAKAYA for a careful bug report)
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>
<code>x</code>	object to be printed
<code>...</code>	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**

```
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"))
## Not run:
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)

```

---

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=NULL, alternative="greater",
  sampling="nonfree", spChk=NULL, adjust.n=TRUE)
## S3 method for class 'jclist'
print(x, ...)

```

## Arguments

<code>fx</code>	a factor of the same length as the neighbours and weights objects in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
<code>sampling</code>	default "nonfree", may be "free"

<code>adjust.n</code>	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 <code>spdep 0.3-28</code> the adjustment was inconsistent - thanks to Tomoki NAKAYA for a careful bug report)
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>
<code>x</code>	object to be printed
<code>...</code>	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:

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

```

jointcount.test(HICRIME, nb2listw(COL.nb, style="C"))
jointcount.test(HICRIME, nb2listw(COL.nb, style="S"))
jointcount.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))
jointcount.test(HICRIME, nb2listw(COL.k4.nb, style="B"))
cat("Note non-symmetric weights matrix - use listw2U()\n")
jointcount.test(HICRIME, listw2U(nb2listw(COL.k4.nb, style="B")))

```

---

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, RANN=TRUE)
```

## Arguments

<code>x</code>	matrix of point coordinates or a <code>SpatialPoints</code> object
<code>k</code>	number of nearest neighbours to be returned
<code>longlat</code>	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if <code>x</code> is a <code>SpatialPoints</code> object, the value is taken from the object itself; <code>longlat</code> will override <code>RANN</code>
<code>RANN</code>	logical value, if the <code>RANN</code> package is available, use for finding $k$ nearest neighbours when <code>longlat</code> is FALSE, and when there are no identical points; from <a href="https://github.com/r-spatial/spdep/issues/38">https://github.com/r-spatial/spdep/issues/38</a> , the input data may have more than two columns if <code>RANN</code> is used

## Details

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

## Value

A list of class `knn`

<code>nn</code>	integer matrix of region number ids
<code>np</code>	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](#), [nn2](#)

**Examples**

```
columbus <- st_read(system.file("shapes/columbus.shp", 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)
summary(gck4.nb, xy, longlat=TRUE)

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)
# https://github.com/r-spatial/spdep/issues/38
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	<i>Neighbours list from knn object</i>
--------	--

---

### 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

<code>knn</code>	A knn object returned by <code>knearneigh</code>
<code>row.names</code>	character vector of region ids to be added to the neighbours list as attribute <code>region.id</code> , default <code>seq(1, nrow(x))</code>
<code>sym</code>	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.shp", package="spData"))[1], quiet=TRUE)
coords <- coordinates(as(columbus, "Spatial"))
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")
```

---

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=NULL, 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 NULL, 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=NULL, 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 NULL, 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 Virgilio Gómez-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=NULL, alternative="greater",
na.action=na.fail, spChk=NULL, return_boot=FALSE)
```

**Arguments**

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



**Value**

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

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

**Author(s)**

Roger Bivand, Virgilio Gómez-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="less")

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

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

---

<code>lee.test</code>	<i>Lee's L test for spatial autocorrelation</i>
-----------------------	---

---

### 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=NULL,
         alternative="greater", na.action=na.fail, spChk=NULL)
```

### Arguments

<code>x</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>y</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of <code>greater</code> (default), <code>less</code> or <code>two.sided</code> .
<code>na.action</code>	a function (default <code>na.fail</code> ), can also be <code>na.omit</code> or <code>na.exclude</code> - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set <code>zero.policy</code> to <code>TRUE</code> because this subsetting may create no-neighbour observations. Note that only weights lists created without using the <code>glist</code> argument to <code>nb2listw</code> may be subsetted. If <code>na.pass</code> is used, zero is substituted for <code>NA</code> values in calculating the spatial lag
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, <code>TRUE</code> , or <code>FALSE</code> , default <code>NULL</code> to use <code>get.spChkOption()</code>

### Value

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

<code>statistic</code>	the value of the standard deviate of Lee's L.
<code>p.value</code>	the p-value of the test.
<code>estimate</code>	the value of the observed Lee's L, its expectation and variance under the method assumption.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the assumption used for calculating the standard deviate.
<code>data.name</code>	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 Gómez-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)
```

---

listw2sn	<i>Spatial neighbour sparse representation</i>
----------	--

---

**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)
```

**Arguments**

<code>listw</code>	a listw object from for example <code>nb2listw</code>
<code>sn</code>	a <code>spatial.neighbour</code> object

**Value**

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

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

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**See Also**

[nb2listw](#)

**Examples**

```
columbus <- st_read(system.file("shapes/columbus.shp", 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

*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 LM test for error dependence (LMerr), the simple LM test for a missing spatially lagged dependent variable (LMlag), variants of these robust to the presence of the other (RLMerr, RLMlag - RLMerr tests for error dependence in the possible presence of a missing lagged dependent variable, RLMlag the other way round), and a portmanteau test (SARMA, in fact LMerr + RLMlag). Note: from spdep 0.3-32, the value of the weights matrix trace term is returned correctly for both underlying symmetric and asymmetric neighbour lists, before 0.3-32, the value was wrong for listw objects based on asymmetric neighbour lists, such as k-nearest neighbours (thanks to Luc Anselin for finding the bug).

**Usage**

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

**Arguments**

<code>model</code>	an object of class <code>lm</code> returned by <code>lm</code> , or optionally a vector of externally calculated residuals (run though <code>na.omit</code> if any NAs present) for use when only "LMerr" is chosen; weights and offsets should not be used in the <code>lm</code> object
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code> , expected to be row-standardised (W-style)
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
<code>test</code>	a character vector of tests requested chosen from <code>LMerr</code> , <code>LMlag</code> , <code>RLMerr</code> , <code>RLMlag</code> , <code>SARMA</code> ; <code>test="all"</code> 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 <code>get.spChkOption()</code>
naSubset	default TRUE to subset listw object for omitted observations in model object (this is a change from earlier behaviour, when the <code>model\$na.action</code> 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 <a href="#">p.adjust</a> ) 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  $\rho$  and spatial error  $\lambda$ :

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

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

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 `LMtestlist` of `hctest` 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.

### 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.

### See Also

[lm](#)

**Examples**

```

data(oldcol)
oldcrime.lm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD)
summary(oldcrime.lm)
res <- lm.LMtests(oldcrime.lm, nb2listw(COL.nb), test=c("LMerr", "LMlag",
  "RLMerr", "RLMlag", "SARMA"))
summary(res)
lm.LMtests(oldcrime.lm, nb2listw(COL.nb))
lm.LMtests(residuals(oldcrime.lm), nb2listw(COL.nb))

```

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()`). The helper function `listw2U()` constructs a weights list object corresponding to the sparse matrix  $\frac{1}{2}(\mathbf{W} + \mathbf{W}')$

**Usage**

```

lm.morantest(model, listw, zero.policy=NULL, alternative = "greater",
  spChk=NULL, resfun=weighted.residuals, naSubset=TRUE)
listw2U(listw)

```

**Arguments**

<code>model</code>	an object of class <code>lm</code> returned by <code>lm</code> ; weights may be specified in the <code>lm</code> fit, but offsets should not be used
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, <code>TRUE</code> , or <code>FALSE</code> , default <code>NULL</code> to use <code>get.spChkOption()</code>
<code>resfun</code>	default: <code>weighted.residuals</code> ; the function to be used to extract residuals from the <code>lm</code> object, may be <code>residuals</code> , <code>weighted.residuals</code> , <code>rstandard</code> , or <code>rstudent</code>
<code>naSubset</code>	default <code>TRUE</code> to subset <code>listw</code> object for omitted observations in model object (this is a change from earlier behaviour, when the <code>model\$na.action</code> component was ignored, and the <code>listw</code> object had to be subsetted by hand)

**Value**

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

<code>statistic</code>	the value of the standard deviate of Moran's I.
<code>p.value</code>	the p-value of the test.
<code>estimate</code>	the value of the observed Moran's I, its expectation and variance under the method assumption.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the method used.
<code>data.name</code>	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 = NULL, 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 <code>lm</code> returned by <code>lm</code> ; weights may be specified in the <code>lm</code> fit, but offsets should not be used
listw	a <code>listw</code> object created for example by <code>nb2listw</code>
zero.policy	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
alternative	a character string specifying the alternative hypothesis, must be one of <code>greater</code> (default), <code>less</code> or <code>two.sided</code> .
spChk	should the data vector names be checked against the spatial objects for identity integrity, <code>TRUE</code> , or <code>FALSE</code> , default <code>NULL</code> to use <code>get.spChkOption()</code>
resfun	default: <code>weighted.residuals</code> ; the function to be used to extract residuals from the <code>lm</code> object, may be <code>residuals</code> , <code>weighted.residuals</code> , <code>rstandard</code> , or <code>rstudent</code>
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 <code>Omega &lt;- invIrW(listw, rho=0.1)</code> ; <code>Omega &lt;- tcrossprod(Omega), chol()</code> is taken internally
save.M	return the full <code>M</code> matrix for use in <code>spdep:::exactMoranAlt</code>
save.U	return the full <code>U</code> matrix for use in <code>spdep:::exactMoranAlt</code>
useTP	default <code>FALSE</code> , if <code>TRUE</code> , use truncation point in integration rather than <code>upper=Inf</code> , see Tiefelsdorf (2000), eq. 6.7, p.69
truncErr	when <code>useTP=TRUE</code> , pass truncation error to truncation point function
zeroTreat	when <code>useTP=TRUE</code> , pass zero adjustment to truncation point function
x	a <code>moranex</code> object
...	arguments to be passed through

**Value**

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

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

**Author(s)**

Markus Reeder and Roger Bivand

**References**

Roger Bivand, Werner G. Müller and Markus Reeder (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.shp", package="spData"))[1])
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
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=NULL, 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 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
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 <code>uniroot</code>
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 <https://doi.org/10.1007/s11749-018-0599-x>

### See Also

[lm.morantest](#)

### Examples

```
eire <- st_read(system.file("shapes/eire.shp", package="spData"))[1]
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
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)
```

---

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 possibility 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=NULL, spChk=NULL, return_internals=FALSE, GeoDa=FALSE)
```

### Arguments

<code>x</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, <code>TRUE</code> , or <code>FALSE</code> , default <code>NULL</code> to use <code>get.spChkOption()</code>
<code>return_internals</code>	default <code>FALSE</code> , if <code>TRUE</code> , return internal values of <i>G</i> , <i>EI</i> and <i>VG</i> as as attribute matrix
<code>GeoDa</code>	default <code>FALSE</code> , if <code>TRUE</code> , drop <code>x</code> values for no-neighbour and self-neighbour only observations from all summations

### 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 can 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 values, with attributes "gstari" set to TRUE or FALSE, "call" set to the function call, and class "localG".

**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 <https://doi.org/10.1007/s11749-018-0599-x>

**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]
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")))

```

---

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 are given in Anselin (1995).

## Usage

```

localmoran(x, listw, zero.policy=NULL, na.action=na.fail,
alternative = "greater", p.adjust.method="none", mlvar=TRUE,
spChk=NULL, adjust.x=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
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.*)

alternative	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
p.adjust.method	a character string specifying the probability value adjustment for multiple tests, default "none"; see <code>p.adjustSP</code> . 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.
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 <code>get.spChkOption()</code>
adjust.x	default FALSE, if TRUE, x values of observations with no neighbours are omitted in the mean of x

### 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), equation 5 p. 334 and A4\*, p. 351. By default, the implementation divides by n, not (n-1) in calculating the variance and higher moments.

### Value

Ii	local moran statistic
E.Ii	expectation of local moran statistic
Var.Ii	variance of local moran statistic
Z.Ii	standard deviate of local moran statistic
Pr()	p-value of local moran statistic

### 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 <https://doi.org/10.1007/s11749-018-0599-x>



**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$totcon, nb2listw(paper.nb),
  p.adjust.method="bonferroni")
printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]),
  check.names=FALSE)
hist(resI[,5])
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)

```

---

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 = "greater", 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 = "greater", 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 <code>lm</code> returned by <code>lm</code> (assuming no global spatial autocorrelation), or an object of class <code>sarlm</code> returned by a spatial simultaneous autoregressive model fit (assuming global spatial autocorrelation represented by the model spatial coefficient); weights may be specified in the <code>lm</code> 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 <code>nb</code>
glist	a list of general weights corresponding to neighbours
style	can take values <code>W</code> , <code>B</code> , <code>C</code> , and <code>S</code>
zero.policy	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
alternative	a character string specifying the alternative hypothesis, must be one of <code>greater</code> (default), <code>less</code> or <code>two.sided</code> .
spChk	should the data vector names be checked against the spatial objects for identity integrity, <code>TRUE</code> , or <code>FALSE</code> , default <code>NULL</code> to use <code>get.spChkOption()</code>
resfun	default: <code>weighted.residuals</code> ; the function to be used to extract residuals from the <code>lm</code> object, may be <code>residuals</code> , <code>weighted.residuals</code> , <code>rstandard</code> , or <code>rstudent</code>
Omega	A SAR process matrix may be passed in to test an alternative hypothesis, for example <code>Omega &lt;- invIrW(listw, rho=0.1)</code> ; <code>Omega &lt;- tcrossprod(Omega), chol()</code> is taken internally
save.Vi	if <code>TRUE</code> , return the star-shaped weights lists for each zone tested
save.M	if <code>TRUE</code> , save a list of left and right <code>M</code> products
useTP	default <code>FALSE</code> , if <code>TRUE</code> , use truncation point in integration rather than <code>upper=Inf</code> , see Tiefelsdorf (2000), eq. 6.7, p.69
truncErr	when <code>useTP=TRUE</code> , pass truncation error to truncation point function
zeroTreat	when <code>useTP=TRUE</code> , pass zero adjustment to truncation point function
x	object to be printed
row.names	ignored argument to <code>as.data.frame.localmoranex</code> ; row names assigned from <code>localmoranex</code> object
optional	ignored argument to <code>as.data.frame.localmoranex</code> ; row names assigned from <code>localmoranex</code> object
...	arguments to be passed through

**Value**

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

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

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

**Author(s)**

Markus Reeder and Roger Bivand

**References**

Bivand RS, Müller W, Reeder 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 <https://doi.org/10.1007/s11749-018-0599-x>

**See Also**

[lm.morantest.exact](#), [localmoran.sad](#)

**Examples**

```
eire <- st_read(system.file("shapes/eire.shp", package="spData"))[1]
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
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$stary ~ e.errorsar$starX - 1)
localmoran.exact.alt(lm.target, nb=eire.nb)
}
if (run) {
Omega <- 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="greater", 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=NULL)

```

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

## 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:

<code>statistic</code>	the value of the saddlepoint approximation of the standard deviate of local Moran's Ii.
<code>p.value</code>	the p-value of the test.
<code>estimate</code>	the value of the observed local Moran's Ii.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the method used.
<code>data.name</code>	a character string giving the name(s) of the data.
<code>internal1</code>	Saddlepoint omega, r and u
<code>df</code>	degrees of freedom
<code>tau</code>	maximum and minimum analytical eigenvalues
<code>i</code>	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.shp", package="spData")[1])
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
eire.nb <- poly2nb(eire)
lw <- nb2listw(eire.nb)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
e.locmor <- summary(localmoransad(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(localmoransad(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)
spatialreg::print.sarlm(e.errorsar)
}
if (run) {
lm.target <- lm(e.errorsar$stary ~ e.errorsar$starX - 1)
Omega <- tcrossprod(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)"])
}

```

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=NULL, na.action=na.fail, spChk=NULL)
```

**Arguments**

<code>x</code>	a numeric vector of the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>a</code>	the exponent applied to the local residuals; the default value of 2 leads to a measure of heterogeneity in the spatial variance

<code>var_hi</code>	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
<code>zero.policy</code>	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
<code>na.action</code>	a function (default <code>na.fail</code> ), can also be <code>na.omit</code> or <code>na.exclude</code> - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set <code>zero.policy</code> to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the <code>glist</code> argument to <code>nb2listw</code> may be subsetted. If <code>na.pass</code> is used, zero is substituted for NA values in calculating the spatial lag. (Note that <code>na.exclude</code> will only work properly starting from R 1.9.0, <code>na.omit</code> and <code>na.exclude</code> assign the wrong classes in 1.8.*)
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>

### 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

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

### Author(s)

Rene Westerholt <westerholt@uni-heidelberg.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 = NULL, na.action = na.fail,
        p.adjust.method = "none", spChk = NULL)
```

**Arguments**

<code>x</code>	a numeric vector of the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>TRUE</code> assign zero to the lagged value of zones without neighbours, if <code>FALSE</code> assign <code>NA</code>
<code>na.action</code>	a function (default <code>na.fail</code> ), can also be <code>na.omit</code> or <code>na.exclude</code> - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set <code>zero.policy</code> to <code>TRUE</code> because this subsetting may create no-neighbour observations. Note that only weights lists created without using the <code>glist</code> argument to <code>nb2listw</code> may be subsetted. If <code>na.pass</code> is used, zero is substituted for <code>NA</code> values in calculating the spatial lag. (Note that <code>na.exclude</code> will only work properly starting from R 1.9.0, <code>na.omit</code> and <code>na.exclude</code> assign the wrong classes in 1.8.*)
<code>p.adjust.method</code>	a character string specifying the probability value adjustment for multiple tests, default "none"; see <a href="#">p.adjustSP</a> . 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.
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, <code>TRUE</code> , or <code>FALSE</code> , default <code>NULL</code> to use <code>get.spChkOption()</code>

**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
$\bar{x}_i$	local spatially weighted mean values
$e_i$	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)**

Rene Westerholt <westerholt@uni-heidelberg.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 = NULL, 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 NULL, 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)**

Rene Westerholt <westerholt@uni-heidelberg.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$CRIME, nb2listw(col.gal.nb), 2, 100)
resLOSH_cs <- LOSH.cs(columbus$CRIME, nb2listw(col.gal.nb))
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 rebuilding the weights list object internally.

## Usage

```
mat2listw(x, row.names = NULL, style="M")
```

## 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 "M", unknown style; if not "M", passed to <a href="#">nb2listw</a> to re-build the object

## 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.shp", package="spData"))[1], quiet=TRUE)
col005 <- dnearneigh(st_coordinates(st_centroid(st_geometry(columbus),
  of_largest_polygon=TRUE)), 0, 0.5, as.character(columbus$NEIGNO))
summary(col005)
col005.w.mat <- nb2mat(col005, zero.policy=TRUE)
col005.w.b <- mat2listw(col005.w.mat)
summary(col005.w.b$neighbours)
diffnb(col005, col005.w.b$neighbours)
col005.w.mat.3T <- kronecker(diag(3), col005.w.mat)
col005.w.b.3T <- mat2listw(col005.w.mat.3T, style="W")
summary(col005.w.b.3T$neighbours)
W <- as(nb2listw(col005, style="W", zero.policy=TRUE), "CsparseMatrix")
col005.spM <- mat2listw(W)
summary(col005.spM$neighbours)
diffnb(col005, col005.spM$neighbours)
IW <- kronecker(Matrix::Diagonal(3), W)
col005.spM.3T <- mat2listw(IW, style="W")
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=NULL, NAOK=FALSE)
```

**Arguments**

<code>x</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>n</code>	number of zones
<code>S0</code>	global sum of weights

zero.policy	default NULL, 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=NULL, alternative="greater",
na.action=na.fail, spChk=NULL, return_boot=FALSE, adjust.n=TRUE)
```

**Arguments**

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

<code>statistic</code>	the value of the observed Moran's I.
<code>parameter</code>	the rank of the observed Moran's I.
<code>p.value</code>	the pseudo p-value of the test.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the method used.
<code>data.name</code>	a character string giving the name(s) of the data, and the number of simulations.
<code>res</code>	<code>nsim</code> 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])

```

---

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, zero.policy=NULL, 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
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()
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))
}
```

---

moran.test	<i>Moran's I test for spatial autocorrelation</i>
------------	---

---

### 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=NULL,
           alternative="greater", rank = FALSE, na.action=na.fail, spChk=NULL,
           adjust.n=TRUE, drop.EI2=FALSE)
```

### Arguments

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

### Value

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

<code>statistic</code>	the value of the standard deviate of Moran's <code>I</code> .
------------------------	---

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 <https://doi.org/10.1007/s11749-018-0599-x>

### 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)

```

---

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 partitioned 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 <code>nb2listw</code> 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 its 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
bh <- st_read(system.file("etc/shapes/bhcv.shp",
  package="spdep")[1], quiet=TRUE)
st_crs(bh) <- "+proj=longlat +ellps=WGS84"
### data padronized
dpad <- data.frame(scale(as.data.frame(bh)[,5:8]))

### neighborhood 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
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, coordinates(as(bh, "Spatial")), 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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
col.tri.nb <- tri2nb(coords)
oldpar <- par(mfrow=c(1,2))
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 known 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 <code>unique(as.character(ID))</code>
ID	identification tags of the locations for the location-less observations; <code>sort(unique(as.character(ID)))</code> must be identical to <code>sort(as.character(attr(nb, "region.id")))</code> ; same length as <code>row.names</code> if provided.
row.names	character vector of observation ids to be added to the neighbours list as attribute <code>region.id</code> , default <code>seq(1, nrow(x))</code> ; 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](#), [dnearest](#), [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**

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

## Usage

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

## Arguments

<code>neighbours</code>	an object of class <code>nb</code>
<code>glist</code>	list of general weights corresponding to neighbours
<code>style</code>	style can take values "W", "B", "C", "U", "minmax" and "S"
<code>zero.policy</code>	default <code>NULL</code> , use global option value; if <code>FALSE</code> stop with error for any empty neighbour sets, if <code>TRUE</code> 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 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 arbitrary 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:

<code>style</code>	one of W, B, C, U, S, minmax as above
<code>neighbours</code>	the input neighbours list
<code>weights</code>	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)

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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
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]]]

```

---

 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 arbitrary 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.shp", 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 converging 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 <a href="#">poly2nb</a> for details.
data	A matrix with observations in the nodes.
id	Node index to compute the cost
id.neigh	Index 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 <a href="#">dist</a> 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 or a SpatialPoints 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

**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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
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)) run <- TRUE
if (run) {
W <- as(nb2listw(col.gal.nb), "CsparseMatrix")
G <- graph.adjacency(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 <- shortest.paths(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, <https://spatial.uchicago.edu/sample-data>.

**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**

<code>p</code>	vector of p-values
<code>nb</code>	a list of neighbours of class <code>nb</code>
<code>method</code>	correction method as defined in <code>p.adjust</code> : "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 plotted.
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	<i>Plot a neighbours list</i>
---------	-------------------------------

---

### 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

<code>x</code>	an object of class <code>nb</code> or (for <code>plot.listw</code> ) class <code>listw</code>
<code>coords</code>	matrix of region point coordinates, a <code>Spatial</code> object (points or polygons), or an <code>sfc</code> object (points or polygons)
<code>col</code>	plotting colour
<code>points</code>	(logical) add points to plot
<code>add</code>	(logical) add to existing plot
<code>arrows</code>	(logical) draw arrowheads for asymmetric neighbours
<code>length</code>	length in plot inches of arrow heads drawn for asymmetric neighbours lists
<code>xlim, ylim</code>	plot window bounds
<code>...</code>	further graphical parameters as in <code>par(...)</code>

### Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

### See Also

[summary.nb](#)

### Examples

```
columbus <- st_read(system.file("shapes/columbus.shp", 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 columns 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=sqrt(.Machine$double.eps),
        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
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, possibly parallelised if a <b>snow</b> cluster is available, otherwise a list of length (n-1) with integer vectors of candidate neighbours (j > i), or NULL if all candidates were (j < i) (as created by the poly_findInBoxGEOS function in <b>rgeos</b> 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.



**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.shp", 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.shp", package="spData")[1], quiet=TRUE)
system.time(xxn <- poly2nb(nc.sids))
system.time(xxn <- poly2nb(as(nc.sids, "Spatial")))
plot(st_geometry(nc.sids))
plot(xxn, coordinates(as(nc.sids, "Spatial")), add=TRUE, col="blue")
```

**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: probability 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.shp", 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 squared differences between the observations 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 columns 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 <a href="#">dist</a> 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 Minimum Spanning Tree*

---

### Description

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

### Usage

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

### Arguments

edges            A matrix with two columns 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. Default 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**

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

**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)
```

**Arguments**

file	name of file with weights data
region.id	region IDs
sn	a <code>spatial.neighbour</code> 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 <code>region.id</code> attribute ID key tags to output file (use in OpenGeoDa will depend on the shapefile having the field named in the <code>ind</code> argument matching the exported tags)
legacy	default FALSE; if TRUE, header has single field with number of observations only

**Details**

Attempts to honour the `region.id` argument given when reading GWT files. If the region IDs given in `region.id=` do not match the origins or destinations in the GWT file, an error will be thrown reporting `Error: !anyNA(reg*dij) is not TRUE` where `*` may be `'o'` for origins or `'d'` for destinations.

**Value**

`read.gwt2nb` returns a neighbour "nb" object with the generalised weights stored as a list element called "dlist" of the "GeoDa" attribute.

**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 <http://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)

```

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)
```

---

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 <b>parallel</b>

## 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)
library(parallel)
nc <- detectCores(logical=FALSE)
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("aple.mc"))
  if(.Platform$OS.type == "windows") {
# forking not permitted on Windows - start cluster
    print(get.mcOption())
    cl <- makeCluster(get.coresOption())
    print(clusterEvalQ(cl, exists("aple.mc")))
    set.ClusterOption(cl)
    clusterEvalQ(get.ClusterOption(), library(spdep))
    print(clusterEvalQ(cl, exists("aple.mc")))
    clusterEvalQ(get.ClusterOption(), detach(package:spdep))
    set.ClusterOption(NULL)
    print(clusterEvalQ(cl, exists("aple.mc")))
    stopCluster(cl)
  }
}
```

```

} else {
  mcOpt <- get.mcOption()
  print(mcOpt)
  print(mclapply(1:get.coresOption(), function(i) exists("aple.mc"),
    mc.cores=get.coresOption()))
  invisible(set.mcOption(FALSE))
  cl <- makeCluster(nc)
  print(clusterEvalQ(cl, exists("aple.mc")))
  set.ClusterOption(cl)
  clusterEvalQ(get.ClusterOption(), library(spdep))
  print(clusterEvalQ(cl, exists("aple.mc")))
  clusterEvalQ(get.ClusterOption(), detach(package:spdep))
  set.ClusterOption(NULL)
  print(clusterEvalQ(cl, exists("aple.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 providing access to a running cluster for embarrassingly parallel tasks.

### Usage

```

set.spChkOption(check)
get.spChkOption()
chkIDs(x, listw)
spNamedVec(var, data)
set.VerboseOption(check)
get.VerboseOption()
set.ZeroPolicyOption(check)
get.ZeroPolicyOption()
set.listw_is_CsparseMatrix_Option(check)
get.listw_is_CsparseMatrix_Option()

```

### Arguments

check	a logical value, TRUE or FALSE
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'cluster Analysis by Tree Edge Removal*


---

## Description

This function implements a SKATER procedure for spatial clustering analysis. This procedure essentially 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 columns 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 <a href="#">dist</a> 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
bh <- st_read(system.file("etc/shapes/bhcv.shp",
  package="spdep")[1], quiet=TRUE)
st_crs(bh) <- "+proj=longlat +ellps=WGS84"
### data standardized
dpad <- data.frame(scale(as.data.frame(bh)[,5:8]))

### neighborhood 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))
plot(mst.bh, coordinates(as(bh, "Spatial")), 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, coordinates(as(bh, "Spatial")), cex.circles=0.035, cex.lab=.7)

### the skater plot, using other colors
plot(res1, coordinates(as(bh, "Spatial")), cex.circles=0.035, cex.lab=.7,
     groups.colors=heat.colors(length(res1$edg)))

### 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 <- detectCores(logical=FALSE)
# 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	<i>Spatial correlogram</i>
----------------	----------------------------

---

**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 number 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.shp", 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 = NULL,
  alternative = "greater", spChk=NULL, return_boot=FALSE)
## S3 method for class 'mc.sim'
plot(x, xlim, xlab, main, sub, ..., ptype="density")
```

**Arguments**

<code>var</code>	a numeric vector the same length as the neighbours list in <code>listw</code>
<code>listw</code>	a <code>listw</code> object created for example by <code>nb2listw</code>
<code>nsim</code>	number of permutations
<code>type</code>	"moran", "geary" or "sokal" criteria for similarity
<code>zero.policy</code>	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
<code>alternative</code>	a character string specifying the alternative hypothesis, must be one of "greater" (default), or "less".
<code>spChk</code>	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use <code>get.spChkOption()</code>
<code>return_boot</code>	return an object of class <code>boot</code> from the equivalent permutation bootstrap rather than an object of class <code>htest</code>
<code>x</code>	the object to be plotted
<code>xlim</code>	the range of the x axis
<code>xlab</code>	a title for the x axis
<code>main</code>	an overall title for the plot
<code>sub</code>	a sub title for the plot
<code>ptype</code>	either "density" or "hist"
<code>...</code>	further arguments passed through

**Value**

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

<code>statistic</code>	the value of the observed Geary's C.
<code>parameter</code>	the rank of the observed Geary's C.
<code>alternative</code>	a character string describing the alternative hypothesis.
<code>method</code>	a character string giving the method used.
<code>data.name</code>	a character string giving the name(s) of the data, and the number of simulations.
<code>p.value</code>	the pseudo p-value of the test.
<code>res</code>	<code>nsim</code> simulated values of statistic, final value is observed statistic
<code>estimate</code>	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$CRIME, nb2listw(COL.nb),
  nsim=99, type="geary", alternative="less")
sim1
plot(sim1)
sp.mantel.mc(COL.OLD$CRIME, nb2listw(COL.nb), nsim=99,
  type="sokal", alternative="less")
sp.mantel.mc(COL.OLD$CRIME, nb2listw(COL.nb), nsim=99,
  type="moran")
```

---

spdep	<i>Return package version number</i>
-------	--------------------------------------

---

**Description**

The function retrieves 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-deprecated	<i>Deprecated Functions in Package <b>spdep</b></i>
------------------	---

---

**Description**

These functions are provided for compatibility with older versions of **spdep** only, and may be defunct as soon as the next release. The functions have been moved to the **spatialreg** package.

**Usage**

```
lextrB(lw, zero.policy = TRUE, control = list())
lextrW(lw, zero.policy=TRUE, control=list())
lextrS(lw, zero.policy=TRUE, control=list())
#l_max(lw, zero.policy=TRUE, control=list())
griffith_sone(P, Q, type="rook")
subgraph_eigenw(nb, glist=NULL, style="W", zero.policy=NULL, quiet=NULL)
mom_calc(lw, m)
mom_calc_int2(is, m, nb, weights, Card)
stsls(formula, data = list(), listw, zero.policy = NULL,
      na.action = na.fail, robust = FALSE, HC=NULL, legacy=FALSE, W2X = TRUE)
## S3 method for class 'stsls'
impacts(obj, ..., tr, R = NULL, listw = NULL, evalues=NULL,
      tol = 1e-06, empirical = FALSE, Q=NULL)
```

```

GMerrorsar(formula, data = list(), listw, na.action = na.fail,
  zero.policy = NULL, method="nlminb", arnoldWied=FALSE,
  control = list(), pars=NULL, scaleU=FALSE, verbose=NULL, legacy=FALSE,
  se.lambda=TRUE, returnHcov=FALSE, pWOrder=250, tol.Hcov=1.0e-10)
## S3 method for class 'gmsar'
summary(object, correlation = FALSE, Hausman=FALSE, ...)
GMargminImage(obj, lambdaseq, s2seq)
gstsls(formula, data = list(), listw, listw2 = NULL, na.action = na.fail,
  zero.policy = NULL, pars=NULL, scaleU=FALSE, control = list(),
  verbose=NULL, method="nlminb", robust=FALSE, legacy=FALSE, W2X=TRUE)
## S3 method for class 'gmsar'
impacts(obj, ..., n = NULL, tr = NULL, R = NULL,
  listw = NULL, evaluates=NULL, tol = 1e-06, empirical = FALSE, Q=NULL)
## S3 method for class 'gmsar'
Hausman.test(object, ..., tol=NULL)
lagmess(formula, data = list(), listw, zero.policy = NULL, na.action = na.fail,
  q = 10, start = -2.5, control=list(), method="BFGS", verbose=NULL,
  use_expm=FALSE)
ME(formula, data=list(), family = gaussian, weights, offset,
  na.action=na.fail, listw=NULL, alpha=0.05, nsim=99, verbose=NULL,
  stdev=FALSE, zero.policy = NULL)
SpatialFiltering(formula, lagformula=NULL, data=list(), na.action=na.fail,
  nb=NULL, glist = NULL, style = "C", zero.policy = NULL, tol = 0.1,
  zerovalue = 1e-04, ExactEV = FALSE, symmetric = TRUE, alpha=NULL,
  alternative="two.sided", verbose=NULL)
LR.sarlm(x, y)
## S3 method for class 'sarlm'
logLik(object, ...)
LR1.sarlm(object)
Wald1.sarlm(object)
## S3 method for class 'sarlm'
Hausman.test(object, ..., tol=NULL)
as.spam.listw(listw)
as_dgRMatrix_listw(listw)
as_dsTMatrix_listw(listw)
as_dsCMatrix_I(n)
as_dsCMatrix_IrW(W, rho)
Jacobian_W(W, rho)
powerWeights(W, rho, order=250, X, tol=.Machine$double.eps^(3/5))
## S3 method for class 'lagImpact'
plot(x, ..., choice="direct", trace=FALSE, density=TRUE)
## S3 method for class 'lagImpact'
print(x, ..., reportQ=NULL)
## S3 method for class 'lagImpact'
summary(object, ..., zstats=FALSE, short=FALSE, reportQ=NULL)
## S3 method for class 'lagImpact'
HPDinterval(obj, prob = 0.95, ..., choice="direct")
intImpacts(rho, beta, P, n, mu, Sigma, irho, drop2beta, bnames, interval,

```

```

type, tr, R, listw, evalues, tol, empirical, Q, icept, iicept, p, mess=FALSE,
  samples=NULL, zero_fill = NULL, dvars = NULL)
can.be.simmed(listw)
eigenw(listw, quiet=NULL)
similar.listw(listw)
do_ldet(coef, env, which=1)
jacobianSetup(method, env, con, pre_eig=NULL, trs=NULL, interval=NULL, which=1)
cheb_setup(env, q=5, which=1)
mcdet_setup(env, p=16, m=30, which=1)
eigen_setup(env, which=1)
eigen_pre_setup(env, pre_eig, which=1)
spam_setup(env, pivot="MMD", which=1)
spam_update_setup(env, in_coef=0.1, pivot="MMD", which=1)
Matrix_setup(env, Imult, super=as.logical(NA), which=1)
Matrix_J_setup(env, super=FALSE, which=1)
LU_setup(env, which=1)
LU_prepermute_setup(env, coef=0.1, order=FALSE, which=1)
moments_setup(env, trs=NULL, m, p, type="MC", correct=TRUE, trunc=TRUE, eq7=TRUE, which=1)
SE_classic_setup(env, SE_method="LU", p=16, m=30, nrho=200, interpn=2000,
  interval=c(-1,0.999), SEldet=NULL, which=1)
SE_whichMin_setup(env, SE_method="LU", p=16, m=30, nrho=200, interpn=2000,
  interval=c(-1,0.999), SEldet=NULL, which=1)
SE_interp_setup(env, SE_method="LU", p=16, m=30, nrho=200,
  interval=c(-1,0.999), which=1)
MCMCsamp(object, mcmc = 1L, verbose = NULL, ...)
## S3 method for class 'spautolm'
MCMCsamp(object, mcmc = 1L, verbose = NULL, ...,
  burnin = 0L, scale=1, listw, control = list())
## S3 method for class 'sarlm'
MCMCsamp(object, mcmc = 1L, verbose = NULL, ...,
  burnin=0L, scale=1, listw, listw2=NULL, control=list())
spautolm(formula, data = list(), listw, weights,
  na.action, family = "SAR", method="eigen", verbose = NULL, trs=NULL,
  interval=NULL, zero.policy = NULL, tol.solve=.Machine$double.eps,
  llprof=NULL, control=list())
## S3 method for class 'spautolm'
summary(object, correlation = FALSE, adj.se=FALSE,
  Nagelkerke=FALSE, ...)
spBreg_sac(formula, data = list(), listw, listw2=NULL, na.action,
  Durbin, type, zero.policy=NULL, control=list())
## S3 method for class 'MCMC_sar_g'
impacts(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
## S3 method for class 'MCMC_sem_g'
impacts(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
## S3 method for class 'MCMC_sac_g'
impacts(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
spBreg_err(formula, data = list(), listw, na.action, Durbin, etype,
  zero.policy=NULL, control=list())

```

```

spBreg_lag(formula, data = list(), listw, na.action, Durbin, type,
  zero.policy=NULL, control=list())
## S3 method for class 'SLX'
predict(object, newdata, listw, zero.policy=NULL, ...)
lmSLX(formula, data = list(), listw, na.action, weights=NULL,
  Durbin=TRUE, zero.policy=NULL)
## S3 method for class 'SLX'
impacts(obj, ...)
create_WX(x, listw, zero.policy=NULL, prefix="")
## S3 method for class 'sarlm'
anova(object, ...)
bptest.sarlm(object, varformula=NULL, studentize = TRUE, data=list())
errorsarlm(formula, data=list(), listw, na.action, weights=NULL,
  Durbin, etype, method="eigen", quiet=NULL, zero.policy=NULL,
  interval = NULL, tol.solve=1.0e-10, trs=NULL, control=list())
## S3 method for class 'sarlm'
impacts(obj, ..., tr, R = NULL, listw = NULL, evalues=NULL,
  useHESS = NULL, tol = 1e-06, empirical = FALSE, Q=NULL)
lagsarlm(formula, data = list(), listw,
  na.action, Durbin, type, method="eigen", quiet=NULL,
  zero.policy=NULL, interval=NULL, tol.solve=1.0e-10, trs=NULL,
  control=list())
## S3 method for class 'sarlm'
predict(object, newdata = NULL, listw = NULL, pred.type = "TS", all.data = FALSE,
  zero.policy = NULL, legacy = TRUE, legacy.mixed = FALSE, power = NULL, order = 250,
  tol = .Machine$double.eps^(3/5), spChk = NULL, ...)
## S3 method for class 'sarlm.pred'
print(x, ...)
## S3 method for class 'sarlm.pred'
as.data.frame(x, ...)
## S3 method for class 'sarlm'
residuals(object, ...)
## S3 method for class 'sarlm'
deviance(object, ...)
## S3 method for class 'sarlm'
coef(object, ...)
## S3 method for class 'sarlm'
vcov(object, ...)
## S3 method for class 'sarlm'
fitted(object, ...)
sacsarlm(formula, data = list(), listw, listw2 = NULL, na.action, Durbin, type,
  method = "eigen", quiet = NULL, zero.policy = NULL, tol.solve = 1e-10,
  llprof=NULL, interval1=NULL, interval2=NULL, trs1=NULL, trs2=NULL,
  control = list())
## S3 method for class 'sarlm'
summary(object, correlation = FALSE, Nagelkerke = FALSE, Hausman=FALSE, adj.se=FALSE, ...)
## S3 method for class 'sarlm'
print(x, ...)

```



```
## S3 method for class 'summary.sarlm'
print(x, digits = max(5, .Options$digits - 3),
      signif.stars = FALSE, ...)
trW(W=NULL, m = 30, p = 16, type = "mult", listw=NULL, momentsSymmetry=TRUE)
```

### Arguments

lw	a binary symmetric listw object from, for example, nb2listw with style “B” for lextB, style “W” for lextW and style “S” for lextS; for l_max, the object may be asymmetric and does not have to be binary
zero.policy	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
control	a list of control arguments
quiet	default NULL, use global !verbose option value; set to FALSE for short summary
P	number of columns in the grid (number of units in a horizontal axis direction)
Q	number of rows in the grid (number of units in a vertical axis direction.)
type	“rook” or “queen”
nb	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”
m	The number of powers; must be an even number for ‘type’=“moments” (default changed from 100 to 30 (2010-11-17))
is	(used internally only in mom_calc_int2 for ‘type’=“moments” on a cluster)
weights	(used internally only in mom_calc_int2 for ‘type’=“moments” on a cluster)
Card	(used internally only in mom_calc_int2 for ‘type’=“moments” on a cluster)
formula	a symbolic description of the model to be fit. The details of model specification are given for lm()
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
listw	a listw object created for example by nb2listw
na.action	a function (default na.fail), can also be na.omit or na.exclude with consequences for residuals and fitted values - 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.
robust	default FALSE, if TRUE, apply a heteroskedasticity correction to the coefficients covariances
HC	default NULL, if robust is TRUE, assigned “HC0”, may take values “HC0” or “HC1” for White estimates or MacKinnon-White estimates respectively

legacy	the argument chooses between two implementations of the robustness correction: default FALSE - use the estimate of Omega only in the White consistent estimator of the variance-covariance matrix, if TRUE, use the original implementation which runs a GLS using the estimate of Omega, and yields different coefficient estimates as well - see example below
W2X	default TRUE, if FALSE only WX are used as instruments in the spatial two stage least squares; until release 0.4-60, only WX were used - see example below
obj	A spatial regression object created by <code>lagsarlm</code> , <code>lagmess</code> or by <code>lmSLX</code> ; in <code>HPDinterval.lagImpact</code> , a <code>lagImpact</code> object
...	Arguments passed through to methods in the <b>cod</b> a package
tr	A vector of traces of powers of the spatial weights matrix created using <code>trW</code> , for approximate impact measures; if not given, <code>listw</code> must be given for exact measures (for small to moderate spatial weights matrices); the traces must be for the same spatial weights as were used in fitting the spatial regression, and must be row-standardised
evalues	vector of eigenvalues of spatial weights matrix for impacts calculations
n	defaults to <code>length(obj\$residuals)</code> ; in the method for <code>gmsar</code> objects it may be used in panel settings to compute the impacts for cross-sectional weights only, suggested by Angela Parenti
R	If given, simulations are used to compute distributions for the impact measures, returned as <code>mcmc</code> objects; the objects are used for convenience but are not output by an MCMC process
tol	Argument passed to <code>mvrnorm</code> : tolerance (relative to largest variance) for numerical lack of positive-definiteness in the coefficient covariance matrix
empirical	Argument passed to <code>mvrnorm</code> (default FALSE): if true, the coefficients and their covariance matrix specify the empirical not population mean and covariance matrix
listw2	a <code>listw</code> object created for example by <code>nb2listw</code> , if not given, set to the same spatial weights as the <code>listw</code> argument
pars	starting values for $\lambda$ and $\sigma^2$ for GMM optimisation, if missing (default), approximated from initial 2sls model as the autocorrelation coefficient corrected for weights style and model sigma squared
scaleU	Default FALSE: scale the OLS residuals before computing the moment matrices; only used if the <code>pars</code> argument is missing
method	default "nlminb", or optionally a method passed to <code>optim</code> to use an alternative optimizer
arnoldWied	default FALSE
returnHcov	default FALSE, return the $V_0$ matrix for a spatial Hausman test
tol.Hcov	the tolerance for computing the $V_0$ matrix (default=1.0e-10)
pWOrder	default 250, if <code>returnHcov=TRUE</code> , pass this order to <code>powerWeights</code> as the power series maximum limit
lambdaseq	if given, an increasing sequence of lambda values for gridding
s2seq	if given, an increasing sequence of sigma squared values for gridding

object	gmsar object from GMerrorsar
correlation	logical; (default=FALSE), TRUE not available
Hausman	if TRUE, the results of the Hausman test for error models are reported
se.lambda	default TRUE, use the analytical method described in <a href="http://econweb.umd.edu/~prucha/STATPROG/OLS/deso1s.pdf">http://econweb.umd.edu/~prucha/STATPROG/OLS/deso1s.pdf</a>
verbose	default NULL, use global option value; if TRUE, reports function values during optimization.
q	default 10; number of powers of the spatial weights to use
start	starting value for numerical optimization, should be a small negative number
use_expm	default FALSE; if TRUE use expm: :expAtv instead of a truncated power series of W
family	a description of the error distribution and link function to be used in the model
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting
alpha	used as a stopping rule to choose all eigenvectors up to and including the one with a p-value exceeding alpha
nsim	number of permutations for permutation bootstrap for finding p-values
stdev	if TRUE, p-value calculated from bootstrap permutation standard deviate using pnorm with alternative="greater", if FALSE the Hope-type p-value
lagformula	An extra one-sided formula to be used when a spatial lag representation is desired; the intercept is excluded within the function if present because it is part of the formula argument, but excluding it explicitly in the lagformula argument in the presence of factors generates a collinear model matrix
zerovalue	eigenvectors with eigenvalues of an absolute value smaller than zerovalue will be excluded in eigenvector search
ExactEV	Set ExactEV=TRUE to use exact expectations and variances rather than the expectation and variance of Moran's I from the previous iteration, default FALSE
symmetric	Should the spatial weights matrix be forced to symmetry, default TRUE
alternative	a character string specifying the alternative hypothesis, must be one of greater, less or two.sided (default).
x	a logLik object or an object for which a logLik() function exists
y	a logLik object or an object for which a logLik() function exists
W	a dsTMatrix object created using as_dsTMatrix_listw from a symmetric listw object
rho	spatial regression coefficient
order	Power series maximum limit
X	A numerical matrix
choice	One of three impacts: direct, indirect, or total
trace	Argument passed to plot.mcmc: plot trace plots
density	Argument passed to plot.mcmc: plot density plots

prob	Argument passed to <code>HPDinterval.mcmc</code> : a numeric scalar in the interval (0,1) giving the target probability content of the intervals
beta, mu, Sigma, irho, drop2beta, bnames, interval, irect, irect, p, mess, samples, zero_fill, dvars	internal arguments shared inside <code>impacts</code> methods
reportQ	default NULL; if TRUE and Q given as an argument to <code>impacts</code> , report impact components
zstats	default FALSE, if TRUE, also return z-values and p-values for the impacts based on the simulations
short	default FALSE, if TRUE passed to the <code>print</code> summary method to omit printing of the mcmc summaries
coef	spatial coefficient value
env	environment containing pre-computed objects, fixed after assignment in setup functions
which	default 1; if 2, use second <code>listw</code> object
con	control list passed from model fitting function and parsed in <code>jacobianSetup</code> to set environment variables for method-specific setup
pre_eig	pre-computed eigenvalues of length n
pivot	default "MMD", may also be "RCM" for Cholesky decomposition using <code>spam</code>
in_coef	fill-in initiation coefficient value, default 0.1
Imult	see <a href="#">Cholesky</a> ; numeric scalar which defaults to zero. The matrix that is decomposed is $A+m*I$ where m is the value of <code>Imult</code> and I is the identity matrix of order <code>ncol(A)</code> . Default in calling <code>spdep</code> functions is 2, here it cannot be missing and does not have a default, but is rescaled for binary weights matrices in proportion to the maximum row sum in those calling functions
super	see <a href="#">Cholesky</a> ; logical scalar indicating is a supernodal decomposition should be created. The alternative is a simplicial decomposition. Default in calling <code>spdep</code> functions is FALSE for "Matrix_J" and <code>as.logical(NA)</code> for "Matrix". Setting it to NA leaves the choice to a CHOLMOD-internal heuristic
trs, trs1, trs2	A numeric vector of m traces, as from <code>trW</code>
correct	default TRUE: use Smirnov correction term, see <a href="#">trW</a>
trunc	default TRUE: truncate Smirnov correction term, see <a href="#">trW</a>
eq7	default TRUE use equation 7 in Smirnov and Anselin (2009), if FALSE no unit root correction
SE_method	default "LU", alternatively "MC"; underlying <code>Indet</code> method to use for generating SE toolbox emulation grid
nrho	default 200, number of <code>Indet</code> values in first stage SE toolbox emulation grid
interval1, interval2	default <code>c(-1,0.999)</code> if interval argument NULL, bounds for SE toolbox emulation grid
interp	default 2000, number of <code>Indet</code> values to interpolate in second stage SE toolbox emulation grid

SEIndet	default NULL, used to pass a pre-computed two-column matrix of coefficient values and corresponding interpolated Indet values
mcmc	The number of MCMC iterations after burnin
burnin	The number of burn-in iterations for the sampler
scale	a positive scale parameter
tol.solve	the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to solve() (default=double precision machine tolerance). Errors in solve() may constitute indications of poorly scaled variables: if the variables have scales differing much from the autoregressive coefficient, the values in this matrix may be very different in scale, and inverting such a matrix is analytically possible by definition, but numerically unstable; rescaling the RHS variables alleviates this better than setting tol.solve to a very small value
llprof	default NULL, can either be an integer, to divide the feasible range into llprof points, or a sequence of spatial coefficient values, at which to evaluate the likelihood function
adj.se	if TRUE, adjust the coefficient standard errors for the number of fitted coefficients
Nagelkerke	if TRUE, the Nagelkerke pseudo R-squared is reported
Durbin	default FALSE (spatial lag model); if TRUE, full spatial Durbin model; if a formula object, the subset of explanatory variables to lag
etype	(use the 'Durbin=' argument - retained for backwards compatibility only) default "error", may be set to "emixed" to include the spatially lagged independent variables added to X; when "emixed", the lagged intercept is dropped for spatial weights style "W", that is row-standardised weights, but otherwise included
newdata	data frame in which to predict — if NULL, predictions are for the data on which the model was fitted. Should have row names corresponding to region.id. If row names are exactly the same than the ones used for training, it uses in-sample predictors for forecast. See 'Details'
prefix	default empty string, may be "lag" in some cases
varformula	a formula describing only the potential explanatory variables for the variance (no dependent variable needed). By default the same explanatory variables are taken as in the main regression model
studentize	logical. If set to TRUE Koenker's studentized version of the test statistic will be used.
useHESS, pred.type, all.data, legacy.mixed, power, spChk, digits, signif.stars	other arguments in deprecated functions
momentsSymmetry	default TRUE; assert Smirnov/Anselin symmetry assumption

## Details

Model-fitting functions and functions supporting model fitting are being moved to the **spatialreg** package.

**See Also**[Deprecated](#)**Examples**

```

## Not run:
data(boston, package="spData")
ab.listb <- nb2listw(boston.soi, style="B")
er <- range(eigenw(ab.listb))
er
res_1 <- lexB(ab.listb)
c(res_1)
run <- FALSE
if (require("RSpectra", quietly=TRUE)) run <- TRUE
if (run) {
B <- as(ab.listb, "CsparseMatrix")
eigs(B, k=1, which="SR")$values
}
if (run) {
eigs(B, k=1, which="LR")$values
}
k5 <- knn2nb(knearneigh(boston.utm, k=5))
c(l_max(nb2listw(k5, style="B")))
max(Re(eigenw(nb2listw(k5, style="B"))))
c(l_max(nb2listw(k5, style="C")))
max(Re(eigenw(nb2listw(k5, style="C"))))
ab.listw <- nb2listw(boston.soi, style="W")
er <- range(eigenw(similar.listw(ab.listw)))
er
res_1 <- lexB(ab.listw)
c(res_1)
if (run) {
B <- as(similar.listw(ab.listw), "CsparseMatrix")
eigs(B, k=1, which="SR")$values
}
if (run) {
eigs(B, k=1, which="LR")$values
}
ab.listw <- nb2listw(boston.soi, style="S")
er <- range(eigenw(similar.listw(ab.listw)))
er
res_1 <- lexB(ab.listw)
c(res_1)
if (run) {
B <- as(similar.listw(ab.listw), "CsparseMatrix")
eigs(B, k=1, which="SR")$values
}
if (run) {
eigs(B, k=1, which="LR")$values
}
rg <- cell2nb(ncol=7, nrow=7, type="rook")
rg_eig <- eigenw(nb2listw(rg, style="B"))

```

```

rg_GS <- griffith_sone(P=7, Q=7, type="rook")
all.equal(rg_eig, rg_GS)
# subgraphs
data(oldcol)
crds <- cbind(COL.OLD$X, COL.OLD$Y)
k3 <- knn2nb(knearneigh(crds, k=3))
nc <- n.comp.nb(k3)
nc$nc
table(nc$comp.id)
k3eig <- eigenw(nb2listw(k3, style="W"))
k3eigSG <- subgraph_eigenw(k3, style="W")
all.equal(sort(k3eig), k3eigSG)

data(oldcol)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
summary(COL.lag.eig, correlation=TRUE)
COL.lag.stsls <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
summary(COL.lag.stsls, correlation=TRUE)
COL.lag.stslsW <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb), W2X=FALSE)
summary(COL.lag.stslsW, correlation=TRUE)
COL.lag.stslsR <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb),
robust=TRUE, W2X=FALSE)
summary(COL.lag.stslsR, correlation=TRUE)
COL.lag.stslsRl <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb),
robust=TRUE, legacy=TRUE, W2X=FALSE)
summary(COL.lag.stslsRl, correlation=TRUE)
data(boston, package="spData")
gp2a <- stsls(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) +
AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi))
summary(gp2a)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
listw <- nb2listw(col.gal.nb)
ev <- eigenw(listw)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
lobj1 <- stsls(CRIME ~ INC + HOVAL, columbus, listw)
loobj1 <- impacts(lobj1, R=200, tr=trMatc)
summary(loobj1, zstats=TRUE, short=TRUE)
loobj2 <- impacts(lobj1, R=200, evals=ev)
summary(loobj2, zstats=TRUE, short=TRUE)
library(coda)
HPDinterval(loobj1)
lobj1r <- stsls(CRIME ~ INC + HOVAL, columbus, listw, robust=TRUE)
loobj1r <- impacts(lobj1r, tr=trMatc, R=200)
summary(loobj1r, zstats=TRUE, short=TRUE)
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="W"), method="eigen")
summary(COL.errW.eig, Hausman=TRUE)
COL.errW.GM <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="W"), returnHcov=TRUE)

```

```

summary(COL.errW.GM, Hausman=TRUE)
aa <- GMargminImage(COL.errW.GM)
levs <- quantile(aa$sZ, seq(0, 1, 1/12))
image(aa, breaks=levs, xlab="lambda", ylab="s2")
points(COL.errW.GM$lambda, COL.errW.GM$s2, pch=3, lwd=2)
contour(aa, levels=signif(levs, 4), add=TRUE)
COL.errW.GM1 <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="W"))
summary(COL.errW.GM1)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[1], quiet=TRUE)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
suppressMessages(ID <- as.character(names(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
identical(substr(ID, 2, 10), substr(as.character(nydata$AREAKEY), 2, 10))
nyadjlw <- mat2listw(nyadjmat, as.character(nydata$AREAKEY))
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
esar1f <- spauto1m(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, family="SAR", method="eigen")
summary(esar1f)
esar1gm <- GMerrorsar(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
  data=nydata, listw=listw_NY)
summary(esar1gm)
esar1gm1 <- GMerrorsar(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
  data=nydata, listw=listw_NY, method="Nelder-Mead")
summary(esar1gm1)
data(baltimore, package="spData")
baltimore$AGE <- ifelse(baltimore$AGE < 1, 1, baltimore$AGE)
lw <- nb2listw(knn2nb(knearneigh(cbind(baltimore$X, baltimore$Y), k=7)))
obj1 <- lm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT),
  data=baltimore)
lm.morantest(obj1, lw)
lm.LMtests(obj1, lw, test="all")
system.time(obj2 <- lagmess(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw))
summary(obj2)
system.time(obj2a <- lagmess(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw,
  use_expm=TRUE))
summary(obj2a)
obj3 <- lagsarlm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw)
summary(obj3)
data(boston, package="spData")
lw <- nb2listw(boston.soi)
gp2 <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2)
  + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
  data=boston.c, lw, method="Matrix")
summary(gp2)
gp2a <- lagmess(CMEDV ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2)
  + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
  data=boston.c, lw)
summary(gp2a)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)

```



```

lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL, data=columbus,
  nb=col.gal.nb, style="W", alpha=0.1, verbose=TRUE)
lagcol
lmlag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
anova(lmbase, lmlag)
set.seed(123)
lagcol1 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
  listw=nb2listw(col.gal.nb), alpha=0.1, verbose=TRUE)
lagcol1
lmlag1 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol1), data=columbus)
anova(lmbase, lmlag1)
set.seed(123)
lagcol2 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
  listw=nb2listw(col.gal.nb), alpha=0.1, stdev=TRUE, verbose=TRUE)
lagcol2
lmlag2 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol2), data=columbus)
anova(lmbase, lmlag2)
NA.columbus <- columbus
NA.columbus$CRIME[20:25] <- NA
COL.ME.NA <- ME(CRIME ~ INC + HOVAL, data=NA.columbus, family="gaussian",
  listw=nb2listw(col.gal.nb), alpha=0.1, stdev=TRUE, verbose=TRUE,
  na.action=na.exclude)
COL.ME.NA$na.action
summary(lm(CRIME ~ INC + HOVAL + fitted(COL.ME.NA), data=NA.columbus,
  na.action=na.exclude))
#nc.sids <- st_read(system.file("shapes/sids.shp", 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)
#ncCR85_nb <- read.gal(system.file("weights/ncCR85.gal", package="spData")[1],
# region.id=rn)
#glmbase <- glm(SID74 ~ 1, data=nc.sids, offset=log(BIR74),
# family="poisson")
#set.seed(123)
#MEpois1 <- ME(SID74 ~ 1, data=nc.sids, offset=log(BIR74),
# family="poisson", listw=nb2listw(ncCR85_nb, style="B"), alpha=0.2, verbose=TRUE)
#MEpois1
#glmME <- glm(SID74 ~ 1 + fitted(MEpois1), data=nc.sids, offset=log(BIR74),
# family="poisson")
#anova(glmME, test="Chisq")
#anova(glmbase, glmME, test="Chisq")
data(hopkins, package="spData")
hopkins_part <- hopkins[21:36,36:21]
hopkins_part[which(hopkins_part > 0, arr.ind=TRUE)] <- 1
hopkins.rook.nb <- cell2nb(16, 16, type="rook")
glmbase <- glm(c(hopkins_part) ~ 1, family="binomial")
set.seed(123)
MEbinom1 <- ME(c(hopkins_part) ~ 1, family="binomial",
  listw=nb2listw(hopkins.rook.nb, style="B"), alpha=0.2, verbose=TRUE)
glmME <- glm(c(hopkins_part) ~ 1 + fitted(MEbinom1), family="binomial")
anova(glmME, test="Chisq")
anova(glmbase, glmME, test="Chisq")
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)

```

```

col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)
sarcol <- SpatialFiltering(CRIME ~ INC + HOVAL, data=columbus,
  nb=col.gal.nb, style="W", ExactEV=TRUE)
sarcol
lmsar <- lm(CRIME ~ INC + HOVAL + fitted(sarcol), data=columbus)
lmsar
anova(lmbase, lmsar)
lm.morantest(lmsar, nb2listw(col.gal.nb))
lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL - 1, data=columbus,
  nb=col.gal.nb, style="W")
lagcol
lmlag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
lmlag
anova(lmbase, lmlag)
lm.morantest(lmlag, nb2listw(col.gal.nb))
NA.columbus <- columbus
NA.columbus$CRIME[20:25] <- NA
COL.SF.NA <- SpatialFiltering(CRIME ~ INC + HOVAL, data=NA.columbus,
  nb=col.gal.nb, style="W", na.action=na.exclude)
COL.SF.NA$na.action
summary(lm(CRIME ~ INC + HOVAL + fitted(COL.SF.NA), data=NA.columbus,
  na.action=na.exclude))
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
mixed <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb),
  type="mixed")
error <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
LR.sarlm(mixed, error)
Hausman.test(error)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
col.listw <- nb2listw(col.gal.nb)
if (require("spam", quietly=TRUE)) {
  col.sp <- as.spam.listw(col.listw)
  str(col.sp)
}
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
nyadjlw <- mat2listw(nyadjmat)
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
library(Matrix)
W_C <- as(listw_NY, "CsparseMatrix")
W_R <- as(listw_NY, "RsparseMatrix")
W_S <- as(listw_NY, "symmetricMatrix")
n <- nrow(W_S)
I <- Diagonal(n)
rho <- 0.1
c(determinant(I - rho * W_S, logarithm=TRUE)$modulus)
sum(log(1 - rho * eigenw(listw_NY)))
nW <- - W_S
nChol <- Cholesky(nW, Imult=8)
n * log(rho) + (2 * c(determinant(update(nChol, nW, 1/rho))$modulus))

```

```

nb7rt <- cell2nb(7, 7, torus=TRUE)
x <- matrix(sample(rnorm(500*length(nb7rt))), nrow=length(nb7rt))
W <- as(nb2listw(nb7rt), "CsparseMatrix")
system.time(ee <- powerWeights(W, rho=0.9, X=x))
#all.equal(e, as(ee, "matrix"), check.attributes=FALSE)
system.time(e <- invIrM(nb7rt, rho=0.98, method="solve", feasible=NULL) %*% x)
system.time(ee <- powerWeights(W, rho=0.98, X=x))
str(attr(ee, "internal"))
all.equal(e, as(ee, "matrix"), check.attributes=FALSE)
system.time(ee <- powerWeights(W, rho=0.98, order=1000, X=x))
all.equal(e, as(ee, "matrix"), check.attributes=FALSE)
nb60rt <- cell2nb(60, 60, torus=TRUE)
W <- as(nb2listw(nb60rt), "CsparseMatrix")
set.seed(1)
x <- matrix(rnorm(dim(W)[1]), ncol=1)
system.time(ee <- powerWeights(W, rho=0.3, X=x))
str(as(ee, "matrix"))
obj <- errorsarlm(as(ee, "matrix")[,1] ~ 1, listw=nb2listw(nb60rt), method="Matrix")
coefficients(obj)
data(oldcol)
W.eig <- eigenw(nb2listw(COL.nb, style="W"))
1/range(W.eig)
S.eig <- eigenw(nb2listw(COL.nb, style="S"))
1/range(S.eig)
B.eig <- eigenw(nb2listw(COL.nb, style="B"))
1/range(B.eig)
# cases for intrinsically asymmetric weights
crds <- cbind(COL.OLD$X, COL.OLD$Y)
k3 <- knn2nb(knearneigh(crds, k=3))
is.symmetric.nb(k3)
k3eig <- eigenw(nb2listw(k3, style="W"))
is.complex(k3eig)
rho <- 0.5
Jc <- sum(log(1 - rho * k3eig))
# complex eigenvalue Jacobian
Jc
W <- as(nb2listw(k3, style="W"), "CsparseMatrix")
I <- diag(length(k3))
Jl <- sum(log(abs(diag(slot(lu(I - rho * W), "U")))))
# LU Jacobian equals complex eigenvalue Jacobian
Jl
all.equal(Re(Jc), Jl)
# wrong value if only real part used
Jr <- sum(log(1 - rho * Re(k3eig)))
Jr
all.equal(Jr, Jl)
# construction of Jacobian from complex conjugate pairs (Jan Hauke)
Rev <- Re(k3eig)[which(Im(k3eig) == 0)]
# real eigenvalues
Cev <- k3eig[which(Im(k3eig) != 0)]
pCev <- Cev[Im(Cev) > 0]
# separate complex conjugate pairs
RpCev <- Re(pCev)

```

```

IpCev <- Im(pCev)
# reassemble Jacobian
Jc1 <- sum(log(1 - rho*Rev)) + sum(log((1 - rho * RpCev)^2 + (rho^2)*(IpCev^2)))
all.equal(Re(Jc), Jc1)
# impact of omitted complex part term in real part only Jacobian
Jc2 <- sum(log(1 - rho*Rev)) + sum(log((1 - rho * RpCev)^2))
all.equal(Jr, Jc2)
# trace of asymmetric (WW) and crossprod of complex eigenvalues for APLE
sum(diag(W %** W))
crossprod(k3eig)
# analytical regular grid eigenvalues
run <- FALSE
if (require("RSpectra", quietly=TRUE)) run <- TRUE
if (run) {
  rg <- cell2nb(ncol=7, nrow=7, type="rook")
  B <- as(nb2listw(rg, style="B"), "CsparseMatrix")
  res1 <- eigs(B, k=1, which="LR")$values
  resn <- eigs(B, k=1, which="SR")$values
  print(Re(c(resn, res1)))
}
if (run) {
  rg_eig <- eigenw(nb2listw(rg, style="B"))
  print(all.equal(range(Re(rg_eig)), c(resn, res1)))
}
if (run) {
  lw <- nb2listw(rg, style="W")
  rg_eig <- eigenw(similar.listw(lw))
  print(range(Re(rg_eig)))
}
if (run) {
  W <- as(lw, "CsparseMatrix")
  print(Re(c(eigs(W, k=1, which="SR")$values, eigs(W, k=1, which="LR")$values)))
}
data(oldcol)
COL.W <- nb2listw(COL.nb, style="W")
COL.S <- nb2listw(COL.nb, style="S")
sum(log(1 - 0.5 * eigenw(COL.W)))
sum(log(1 - 0.5 * eigenw(similar.listw(COL.W))))
W_J <- as(as_dsTMatrix_listw(similar.listw(COL.W)), "CsparseMatrix")
I <- as_dsCMatrix_I(dim(W_J)[1])
c(determinant(I - 0.5 * W_J, logarithm=TRUE)$modulus)
sum(log(1 - 0.5 * eigenw(COL.S)))
sum(log(1 - 0.5 * eigenw(similar.listw(COL.S))))
W_J <- as(as_dsTMatrix_listw(similar.listw(COL.S)), "CsparseMatrix")
c(determinant(I - 0.5 * W_J, logarithm=TRUE)$modulus)
data(boston, package="spData")
lw <- nb2listw(boston.soi)
can.sim <- spdep:::can.be.simmed(lw)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("verbose", FALSE, envir=env)

```

```

assign("family", "SAR", envir=env)
eigen_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("verbose", FALSE, envir=env)
assign("family", "SAR", envir=env)
assign("n", length(boston.soi), envir=env)
eigen_pre_setup(env, pre_eig=eigenw(similar.listw(lw)))
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
assign("n", length(boston.soi), envir=env)
Matrix_setup(env, Imult=2, super=FALSE)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
if (require("spam", quietly=TRUE)) {
  env <- new.env(parent=globalenv())
  assign("listw", lw, envir=env)
  assign("n", length(boston.soi), envir=env)
  assign("can.sim", can.sim, envir=env)
  assign("similar", FALSE, envir=env)
  assign("family", "SAR", envir=env)
  spam_setup(env)
  get("similar", envir=env)
  do_ldet(0.5, env)
  rm(env)
}
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
LU_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
LU_prepermute_setup(env)
get("similar", envir=env)

```

```

do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
cheb_setup(env, q=5)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
set.seed(12345)
mcdet_setup(env, p=16, m=30)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[1], quiet=TRUE)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
suppressMessages(ID <- as.character(names(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
  identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEYS), 2, 10))
nyadjlw <- mat2listw(nyadjmat)
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, family="SAR", method="eigen")
summary(esar1f)
res <- MCMCsamp(esar1f, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
#esar1fw <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8, family="SAR", method="eigen")
#summary(esar1fw)
#res <- MCMCsamp(esar1fw, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
ecar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, family="CAR", method="eigen")
summary(ecar1f)
res <- MCMCsamp(ecar1f, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
#esar1fw <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8, family="SAR", method="eigen")
#summary(esar1fw)
#res <- MCMCsamp(esar1fw, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
#ecar1fw <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8, family="CAR", method="eigen")
#summary(ecar1fw)
#res <- MCMCsamp(ecar1fw, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)

```

```

esar0 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY)
summary(esar0)
res <- MCMCsamp(esar0, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
#esar0w <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8)
#summary(esar0w)
#res <- MCMCsamp(esar0w, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
esar1 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, etype="emixed")
summary(esar1)
res <- MCMCsamp(esar1, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
lsar0 <- lagsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY)
summary(lsar0)
res <- MCMCsamp(lsar0, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
lsar1 <- lagsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, type="mixed")
summary(lsar1)
res <- MCMCsamp(lsar1, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
ssar0 <- sacsarlml(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY)
summary(ssar0)
res <- MCMCsamp(ssar0, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
ssar1 <- sacsarlml(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, type="sacmixed")
summary(ssar1)
res <- MCMCsamp(ssar1, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[1], quiet=TRUE)
lm0 <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata)
summary(lm0)
lm0w <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, weights=POP8)
summary(lm0w)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
suppressMessages(ID <- as.character(names(foreign::read.dbf(system.file(
  "misc/nyadjwts.dbf", package="spData")[1])[-1]))
identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEY), 2, 10))
nyadjlw <- mat2listw(nyadjmat, as.character(nydata$AREAKEY))
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
esar0 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY)
summary(esar0)
system.time(esar1f <- spauto1m(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
  data=nydata, listw=listw_NY, family="SAR", method="eigen"))
res <- summary(esar1f)

```

```

print(res)
sqrt(diag(res$resvar))
sqrt(diag(esar1f$fit$imat)*esar1f$fit$s2)
sqrt(diag(esar1f$fdHess))
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
  data=nydata, listw=listw_NY, family="SAR", method="Matrix"))
summary(esar1M)
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
  data=nydata, listw=listw_NY, family="SAR", method="Matrix",
  control=list(super=TRUE)))
summary(esar1M)
#esar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8, family="SAR", method="eigen")
#summary(esar1wf)
#system.time(esar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
# data=nydata, listw=listw_NY, weights=POP8, family="SAR", method="Matrix"))
#summary(esar1wM)
#esar1wlu <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8, family="SAR", method="LU")
#summary(esar1wlu)
#esar1wch <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=POP8, family="SAR", method="Chebyshev")
#summary(esar1wch)
ecar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
  listw=listw_NY, family="CAR", method="eigen")
summary(ecar1f)
system.time(ecar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
  data=nydata, listw=listw_NY, family="CAR", method="Matrix"))
summary(ecar1M)
#ecar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=nydata$POP8, family="CAR", method="eigen")
#summary(ecar1wf)
#system.time(ecar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
# data=nydata, listw=listw_NY, weights=POP8, family="CAR", method="Matrix"))
#summary(ecar1wM)
nc.sids <- st_read(system.file("shapes/sids.shp", package="spData"))[1], quiet=TRUE)
ft.SID74 <- sqrt(1000)*(sqrt(nc.sids$SID74/nc.sids$BIR74) +
  sqrt((nc.sids$SID74+1)/nc.sids$BIR74))
lm_nc <- lm(ft.SID74 ~ 1)
sids.nhbr30 <- dnearneigh(cbind(nc.sids$east, nc.sids$north), 0, 30, row.names=row.names(nc.sids))
sids.nhbr30.dist <- nbdists(sids.nhbr30, cbind(nc.sids$east, nc.sids$north))
sids.nhbr <- listw2sn(nb2listw(sids.nhbr30, glist=sids.nhbr30.dist, style="B", zero.policy=TRUE))
dij <- sids.nhbr[,3]
n <- nc.sids$BIR74
e11 <- min(dij)/dij
e12 <- sqrt(n[sids.nhbr$to]/n[sids.nhbr$from])
sids.nhbr$weights <- e11*e12
sids.nhbr.listw <- sn2listw(sids.nhbr)
both <- factor(paste(nc.sids$L_id, nc.sids$M_id, sep=":"))
ft.NWBIR74 <- sqrt(1000)*(sqrt(nc.sids$NWBIR74/nc.sids$BIR74) +
  sqrt((nc.sids$NWBIR74+1)/nc.sids$BIR74))
mdata <- data.frame(both, ft.NWBIR74, ft.SID74, BIR74=nc.sids$BIR74)
out1 <- which.max(rstandard(lm_nc))

```



```

as.character(nc.sids$NAME[out1])
mdata.4 <- mdata[-out1,]
W <- listw2mat(sids.nhbr.listw)
W.4 <- W[-out1, -out1]
sids.nhbr.listw.4 <- mat2listw(W.4)
esarI <- errorsarlm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
  zero.policy=TRUE)
summary(esarI)
esarIa <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
  family="SAR")
summary(esarIa)
esarIV <- errorsarlm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbr.listw,
  zero.policy=TRUE)
summary(esarIV)
esarIVa <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbr.listw,
  family="SAR")
summary(esarIVa)
#esarIaw <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
# weights=BIR74, family="SAR")
#summary(esarIaw)
#esarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata, listw=sids.nhbr.listw,
# weights=BIR74, family="SAR")
#summary(esarIIaw)
#esarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata,
# listw=sids.nhbr.listw, weights=BIR74, family="SAR")
#summary(esarIVaw)
#ecarIaw <- spautolm(ft.SID74 ~ 1, data=mdata.4, listw=sids.nhbr.listw.4,
# weights=BIR74, family="CAR")
#summary(ecarIaw)
#ecarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata.4,
# listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
#summary(ecarIIaw)
#ecarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata.4,
# listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
#summary(ecarIVaw)
#nc.sids$fitIV <- append(fitted.values(ecarIVaw), NA, out1-1)
#plot(nc.sids[, "fitIV"], nbreaks=12) # Cressie 1993, p. 565
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="W"))
summary(COL.errW.eig)
COL.errW.sar <- spautolm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="W"))
summary(COL.errW.sar)
data(boston, package="spData")
gp1 <- spautolm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2)
  + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
  data=boston.c, nb2listw(boston.soi), family="SMA")
summary(gp1)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
ev <- eigenw(listw)
COL.sacW.eig <- sacsarlml(CRIME ~ INC + HOVAL, data=COL.OLD, listw,

```

```

control=list(pre_eig1=ev, pre_eig2=ev)
summary(COL.sacW.eig)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
set.seed(1)
summary(impacts(COL.sacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
library(coda)
set.seed(1)
COL.sacW.B0 <- spBreg_sac(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  Durbin=FALSE, control=list(ndraw=1500L, nomit=500L))
print(summary(COL.sacW.B0))
print(summary(impacts(COL.sacW.B0, tr=trMatc), zstats=TRUE, short=TRUE))
set.seed(1)
COL.sacW.B1 <- spBreg_sac(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  Durbin=TRUE, control=list(ndraw=1500L, nomit=500L))
print(summary(COL.sacW.B1))
print(summary(impacts(COL.sacW.B1, tr=trMatc), zstats=TRUE, short=TRUE))
COL.msacW.eig <- sacsarlms(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  type="sacmixed", control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW.eig)
set.seed(1)
summary(impacts(COL.msacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW1.eig <- sacsarlms(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  Durbin=TRUE, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW1.eig)
set.seed(1)
summary(impacts(COL.msacW1.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW2.eig <- sacsarlms(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
  listw, Durbin= ~ INC, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW2.eig)
summary(impacts(COL.msacW2.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
library(coda)
set.seed(1)
data(oldcol)
lw <- nb2listw(COL.nb, style="w")
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
  control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
  Durbin=TRUE)
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
  Durbin=TRUE, control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))

```

```

print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
  Durbin=~INC)
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
  Durbin=~INC, control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
COL.lag.Bayes <- spBreg_lag(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw=listw)
summary(COL.lag.Bayes)
summary(impacts(COL.lag.Bayes, tr=trMatc), short=TRUE, zstats=TRUE)
summary(impacts(COL.lag.Bayes, values=ev), short=TRUE, zstats=TRUE)
set.seed(1)
COL.D0.Bayes <- spBreg_lag(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw=listw, Durbin=TRUE)
summary(COL.D0.Bayes)
summary(impacts(COL.D0.Bayes, tr=trMatc), short=TRUE, zstats=TRUE)
set.seed(1)
COL.D1.Bayes <- spBreg_lag(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
  listw=listw, Durbin= ~ INC)
summary(COL.D1.Bayes)
summary(impacts(COL.D1.Bayes, tr=trMatc), short=TRUE, zstats=TRUE)
#data(elect80, package="spData")
#lw <- nb2listw(e80_queen, zero.policy=TRUE)
#el_m1 <- lagsar1m(log(pc_turnout) ~ log(pc_college) + log(pc_homeownership)
# + log(pc_income), data=elect80, listw=lw, zero.policy=TRUE, method="LU")
#summary(el_m1)
#set.seed(1)
#el_B <- spBreg_lag(log(pc_turnout) ~ log(pc_college) + log(pc_homeownership)
# + log(pc_income), data=elect80, listw=lw, zero.policy=TRUE)
#summary(el_B)
#el_m1$timings
#attr(el_B, "timings")
data(oldcol, package="spdep")
lw <- spdep::nb2listw(COL.nb)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
pslx0 <- predict(COL.SLX)
pslx1 <- predict(COL.SLX, newdata=COL.OLD, listw=lw)
all.equal(pslx0, pslx1)
COL.OLD1 <- COL.OLD
COL.OLD1$INC <- COL.OLD1$INC + 1
pslx2 <- predict(COL.SLX, newdata=COL.OLD1, listw=lw)
sum(coef(COL.SLX)[c(2,4)])

```

```

mean(pslx2-pslx1)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
summary(COL.SLX)
summary(impacts(COL.SLX))
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL + I(HOVAL^2), data=COL.OLD, listw=lw, Durbin=TRUE)
summary(impacts(COL.SLX))
summary(COL.SLX)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL + I(HOVAL^2), data=COL.OLD, listw=lw, Durbin=~INC)
summary(impacts(COL.SLX))
summary(COL.SLX)
COL.SLX <- lmSLX(CRIME ~ INC, data=COL.OLD, listw=lw)
summary(COL.SLX)
summary(impacts(COL.SLX))
crds <- cbind(COL.OLD$X, COL.OLD$Y)
mdist <- sqrt(sum(diff(apply(crds, 2, range))^2))
dnb <- dnearneigh(crds, 0, mdist)
dists <- nb2dists(dnb, crds)
f <- function(x, form, data, dnb, dists, verbose) {
  glst <- lapply(dists, function(d) 1/(d^x))
  lw <- nb2listw(dnb, glst=glst, style="B")
  res <- logLik(lmSLX(form=form, data=data, listw=lw))
  if (verbose) cat("power:", x, "logLik:", res, "\n")
  res
}
opt <- optimize(f, interval=c(0.1, 4), form=CRIME ~ INC + HOVAL,
  data=COL.OLD, dnb=dnb, dists=dists, verbose=TRUE, maximum=TRUE)
glst <- lapply(dists, function(d) 1/(d^opt$maximum))
lw <- nb2listw(dnb, glst=glst, style="B")
SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
summary(SLX)
summary(impacts(SLX))
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
# fit models for comparison
lm.mod <- lm(CRIME ~ HOVAL + INC, data=columbus)
lag <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
mixed <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb),
  Durbin=TRUE)
error <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
# compare nested models
LR.sarlm(mixed, error)
#anova(lag, lm.mod)
#anova(lag, error, mixed)
AIC(lag, error, mixed)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
error.col <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus,
  nb2listw(col.gal.nb))
bptest.sarlm(error.col)
bptest.sarlm(error.col, studentize=FALSE)
lm.target <- lm(error.col$tary ~ error.col$tarX - 1)
if (require(lmtest) && require(sandwich)) {
  print(coeftest(lm.target, vcov=vcovHC(lm.target, type="HC0"), df=Inf))
}

```

```

}
data(oldcol)
lw <- nb2listw(COL.nb, style="W")
ev <- eigenw(similar.listw(lw))
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, quiet=FALSE, control=list(pre_eig=ev))
summary(COL.errW.eig)
COL.errW.eig_ev <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, control=list(pre_eig=ev))
all.equal(coefficients(COL.errW.eig), coefficients(COL.errW.eig_ev))
COL.errB.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="B"))
summary(COL.errB.eig)
W <- as(nb2listw(COL.nb), "CsparseMatrix")
trMatc <- trW(W, type="mult")
COL.errW.M <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix", quiet=FALSE, trs=trMatc)
summary(COL.errW.M)
COL.SDEM.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, etype="emixed", control=list(pre_eig=ev))
summary(COL.SDEM.eig)
COL.SDEM.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, Durbin=TRUE, control=list(pre_eig=ev))
summary(COL.SDEM.eig)
COL.SDEM.eig <- errorsarlm(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
  lw, Durbin=~INC, control=list(pre_eig=ev))
summary(COL.SDEM.eig)
summary(impacts(COL.SDEM.eig))
NA.COL.OLD <- COL.OLD
NA.COL.OLD$CRIME[20:25] <- NA
COL.err.NA <- errorsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
  nb2listw(COL.nb), na.action=na.exclude)
COL.err.NA$na.action
COL.err.NA
resid(COL.err.NA)
lw <- nb2listw(COL.nb, style="W")
print(system.time(ev <- eigenw(similar.listw(lw))))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="eigen", control=list(pre_eig=ev))))
ocoef <- coefficients(COL.errW.eig)
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="eigen", control=list(pre_eig=ev, LAPACK=FALSE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="eigen", control=list(pre_eig=ev, compiled_sse=TRUE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix_J", control=list(super=TRUE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix_J", control=list(super=FALSE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,

```

```

  lw, method="Matrix_J", control=list(super=as.logical(NA))))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix", control=list(super=TRUE))))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix", control=list(super=FALSE))))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix", control=list(super=as.logical(NA))))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
if (require("spam", quietly=TRUE)) {
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="spam", control=list(spamPivot="MMD"))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="spam", control=list(spamPivot="RCM"))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="spam_update", control=list(spamPivot="MMD"))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="spam_update", control=list(spamPivot="RCM"))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
}
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
listw <- nb2listw(col.gal.nb)
ev <- eigenw(listw)
lobj <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw,
  control=list(pre_eig=ev))
summary(lobj)
mobj <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, Durbin=TRUE,
  control=list(pre_eig=ev))
summary(mobj)
mobj1 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, Durbin= ~ INC,
  control=list(pre_eig=ev))
summary(mobj1)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
trMC <- trW(W, type="MC")
set.seed(1)
impacts(lobj, listw=listw)
impacts(lobj, tr=trMatc)
impacts(lobj, tr=trMC)
impacts(lobj, evalues=ev)
library(coda)
lobjIQ5 <- impacts(lobj, tr=trMatc, R=200, Q=5)
summary(lobjIQ5, zstats=TRUE, short=TRUE)
summary(lobjIQ5, zstats=TRUE, short=TRUE, reportQ=TRUE)
impacts(mobj, listw=listw)
impacts(mobj, tr=trMatc)
impacts(mobj, tr=trMC)

```

```

impacts(mobj1, tr=trMatc)
impacts(mobj1, listw=listw)
cat(try(impacts(mobj, evalues=ev), silent=TRUE), "\n")
summary(impacts(mobj, tr=trMatc, R=200), short=TRUE, zstats=TRUE)
summary(impacts(mobj1, tr=trMatc, R=200), short=TRUE, zstats=TRUE)
#xobj <- lmSLX(CRIME ~ INC + HOVAL, columbus, listw)
#summary(impacts(xobj))
eobj <- errorsarlm(CRIME ~ INC + HOVAL, columbus, listw, etype="emixed")
summary(impacts(eobj), adjust_k=TRUE)
mobj1 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="Matrix", control=list(fdHess=TRUE))
summary(mobj1)
set.seed(1)
summary(impacts(mobj1, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
summary(impacts(mobj, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
mobj2 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="Matrix", control=list(fdHess=TRUE, optimHess=TRUE))
summary(impacts(mobj2, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
if (require("spam", quietly=TRUE)) {
mobj3 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="spam", control=list(fdHess=TRUE))
summary(impacts(mobj3, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
}
data(boston, package="spData")
Wb <- as(nb2listw(boston.soi), "CsparseMatrix")
trMatb <- trW(Wb, type="mult")
gp2mMi <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed", method="Matrix",
control=list(fdHess=TRUE), trs=trMatb)
summary(gp2mMi)
summary(impacts(gp2mMi, tr=trMatb, R=1000), zstats=TRUE, short=TRUE)
#data(house, package="spData")
#lw <- nb2listw(L0_nb)
#form <- formula(log(price) ~ age + I(age^2) + I(age^3) + log(lotsize) +
# rooms + log(TLA) + beds + syear)
#lobj <- lagsarlm(form, house, lw, method="Matrix",
# control=list(fdHess=TRUE), trs=trMat)
#summary(loobj)
#loobj <- impacts(loobj, tr=trMat, R=1000)
#summary(loobj, zstats=TRUE, short=TRUE)
#loobj1 <- stsls(form, house, lw)
#loobj1 <- impacts(loobj1, tr=trMat, R=1000)
#summary(loobj1, zstats=TRUE, short=TRUE)
#mobj <- lagsarlm(form, house, lw, type="mixed",
# method="Matrix", control=list(fdHess=TRUE), trs=trMat)
#summary(mobj)
#moobj <- impacts(mobj, tr=trMat, R=1000)
#summary(moobj, zstats=TRUE, short=TRUE)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
ev <- eigenw(listw)
W <- as(listw, "CsparseMatrix")

```

```

trMatc <- trW(W, type="mult")
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw,
  method="eigen", quiet=FALSE, control=list(pre_eig=ev, OrdVsign=1))
summary(COL.lag.eig, correlation=TRUE)
COL.lag.eig$fdHess
COL.lag.eig$resvar
# using the apparent sign in Ord (1975, equation B.1)
COL.lag.eigb <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw,
  method="eigen", control=list(pre_eig=ev, OrdVsign=-1))
summary(COL.lag.eigb)
COL.lag.eigb$fdHess
COL.lag.eigb$resvar
# force numerical Hessian
COL.lag.eig1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw=listw, method="Matrix", control=list(small=25))
summary(COL.lag.eig1)
COL.lag.eig1$fdHess
# force LeSage & Pace (2008, p. 57) approximation
COL.lag.eig1a <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw=listw, method="Matrix", control=list(small=25), trs=trMatc)
summary(COL.lag.eig1a)
COL.lag.eig1a$fdHess
COL.lag.eig$resvar[2,2]
# using the apparent sign in Ord (1975, equation B.1)
COL.lag.eigb$resvar[2,2]
# force numerical Hessian
COL.lag.eig1$fdHess[1,1]
# force LeSage & Pace (2008, p. 57) approximation
COL.lag.eig1a$fdHess[2,2]
system.time(COL.lag.M <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb), method="Matrix", quiet=FALSE))
summary(COL.lag.M)
impacts(COL.lag.M, listw=nb2listw(COL.nb))
if (require("spam", quietly=TRUE)) {
system.time(COL.lag.sp <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb), method="spam", quiet=FALSE))
summary(COL.lag.sp)
}
COL.lag.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="B"))
summary(COL.lag.B)
COL.mixed.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="B"), type="mixed", tol.solve=1e-9,
  control=list(pre_eig=ev))
summary(COL.mixed.B)
COL.mixed.W <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, type="mixed",
  control=list(pre_eig=ev))
summary(COL.mixed.W)
COL.mixed.D00 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, Durbin=TRUE,
  control=list(pre_eig=ev))
summary(COL.mixed.D00)

```



```

COL.mixed.D01 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, Durbin=FALSE,
  control=list(pre_eig=ev))
summary(COL.mixed.D01)
COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, Durbin= ~ INC + HOVAL,
  control=list(pre_eig=ev))
summary(COL.mixed.D1)
f <- CRIME ~ INC + HOVAL
COL.mixed.D2 <- lagsarlm(f, data=COL.OLD, listw,
  Durbin=as.formula(delete.response(terms(f))),
  control=list(pre_eig=ev))
summary(COL.mixed.D2)
COL.mixed.D1a <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, Durbin= ~ INC,
  control=list(pre_eig=ev))
summary(COL.mixed.D1a)
try(COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, Durbin= ~ inc + HOVAL,
  control=list(pre_eig=ev)))
try(COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  listw, Durbin= ~ DISCBD + HOVAL,
  control=list(pre_eig=ev)))
NA.COL.OLD <- COL.OLD
NA.COL.OLD$CRIME[20:25] <- NA
COL.lag.NA <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
  nb2listw(COL.nb), na.action=na.exclude,
  control=list(tol.opt=.Machine$double.eps^0.4))
COL.lag.NA$na.action
COL.lag.NA
resid(COL.lag.NA)
data(boston, package="spData")
gp2mM <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
  I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
  data=boston.c, nb2listw(boston soi), type="mixed", method="Matrix")
summary(gp2mM)
W <- as(nb2listw(boston soi), "CsparseMatrix")
trMatb <- trW(W, type="mult")
gp2mMi <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
  I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
  data=boston.c, nb2listw(boston soi), type="mixed", method="Matrix",
  trs=trMatb)
summary(gp2mMi)
data(oldcol)
lw <- nb2listw(COL.nb)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw)

COL.mix.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw,
  type="mixed")
print(p1 <- predict(COL.mix.eig))
#print(p2 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy.mixed = TRUE))
AIC(COL.mix.eig)

```

```

sqrt(deviance(COL.mix.eig)/length(COL.nb))
sqrt(sum((COL.OLD$CRIME - as.vector(p1))^2)/length(COL.nb))
#sqrt(sum((COL.OLD$CRIME - as.vector(p2))^2)/length(COL.nb))

COL.err.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw)
AIC(COL.err.eig)
sqrt(deviance(COL.err.eig)/length(COL.nb))
sqrt(sum((COL.OLD$CRIME - as.vector(predict(COL.err.eig)))^2)/length(COL.nb))
#sqrt(sum((COL.OLD$CRIME - as.vector(predict(COL.err.eig, newdata=COL.OLD,
# listw=lw, pred.type = "TS")))^2)/length(COL.nb))

COL.SDerr.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw,
  etype="emixed")
AIC(COL.SDerr.eig)
sqrt(deviance(COL.SDerr.eig)/length(COL.nb))
sqrt(sum((COL.OLD$CRIME - as.vector(predict(COL.SDerr.eig)))^2)/length(COL.nb))
#sqrt(sum((COL.OLD$CRIME - as.vector(predict(COL.SDerr.eig, newdata=COL.OLD,
# listw=lw, pred.type = "TS")))^2)/length(COL.nb))

AIC(COL.lag.eig)
sqrt(deviance(COL.lag.eig)/length(COL.nb))
sqrt(sum((COL.OLD$CRIME - as.vector(predict(COL.lag.eig)))^2)/length(COL.nb))
#sqrt(sum((COL.OLD$CRIME - as.vector(predict(COL.lag.eig, newdata=COL.OLD,
# listw=lw, pred.type = "TS")))^2)/length(COL.nb))

#p3 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy=FALSE, legacy.mixed = TRUE)
#all.equal(p2, p3, check.attributes=FALSE)
#p4 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy=FALSE, power=TRUE, legacy.mixed = TRUE)
#all.equal(p2, p4, check.attributes=FALSE)
#p5 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy=TRUE, power=TRUE, legacy.mixed = TRUE)
#all.equal(p2, p5, check.attributes=FALSE)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
ev <- eigenw(listw)
COL.sacW.eig <- sacsarlml(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.sacW.eig)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
set.seed(1)
summary(impacts(COL.sacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW.eig <- sacsarlml(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  type="sacmixed", control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW.eig)
set.seed(1)
summary(impacts(COL.msacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW1.eig <- sacsarlml(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
  Durbin=TRUE, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW1.eig)
set.seed(1)

```

```

summary(impacts(COL.msacW1.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW2.eig <- sacsarlml(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
  listw, Durbin= ~ INC, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW2.eig)
summary(impacts(COL.msacW2.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
data(oldcol)
COL.mix.eig <- lagsarlml(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb), type="mixed", method="eigen")
summary(COL.mix.eig, correlation=TRUE, Nagelkerke=TRUE)
COL.mix.M <- lagsarlml(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb), type="mixed", method="Matrix")
summary(COL.mix.M, correlation=TRUE, Nagelkerke=TRUE)
COL.errW.eig <- errorsarlml(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="W"), method="eigen")
summary(COL.errW.eig, correlation=TRUE, Nagelkerke=TRUE, Hausman=TRUE)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
listw <- nb2listw(col.gal.nb)
W <- as(listw, "CsparseMatrix")
system.time(trMat <- trW(W, type="mult"))
str(trMat)
set.seed(1100)
system.time(trMC <- trW(W, type="MC"))
str(trMC)
plot(trMat, trMC)
abline(a=0, b=1)
for(i in 3:length(trMC)) {
  segments(trMat[i], trMC[i]-2*attr(trMC, "sd")[i], trMat[i],
    trMC[i]+2*attr(trMC, "sd")[i])
}
listwS <- similar.listw(listw)
W <- Matrix::forceSymmetric(as(listwS, "CsparseMatrix"))
system.time(trmom <- trW(W, m=24, type="moments"))
str(trmom)
all.equal(trMat[1:24], trmom, check.attributes=FALSE)
system.time(trMat <- trW(W, m=24, type="mult"))
str(trMat)
all.equal(trMat, trmom, check.attributes=FALSE)
set.seed(1)
system.time(trMC <- trW(W, m=24, type="MC"))
str(trMC)
data(boston, package="spData")
listw <- nb2listw(boston.soi)
listwS <- similar.listw(listw)
system.time(trmom <- trW(listw=listwS, m=24, type="moments"))
str(trmom)
library(parallel)
nc <- detectCores(logical=FALSE)
# set nc to 1L here
if (nc > 1L) nc <- 1L
coresOpt <- get.coresOption()
invisible(set.coresOption(nc))
if(!get.mcOption()) {

```

```

    cl <- makeCluster(get.coresOption())
    set.ClusterOption(cl)
  }

  ## End(Not run) #dontrun

```

---

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=NULL, adjust.n=TRUE)
Szero(listw)

```

### Arguments

listw	a listw object from for example nb2listw
zero.policy	default NULL, 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 <a href="#">dist</a> 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 = NULL, ...)
```

**Arguments**

x	an object of class listw
subset	logical expression
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 - 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`.

**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)
```

---

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**

<code>x</code>	an object of class <code>nb</code>
<code>subset</code>	logical expression
<code>...</code>	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.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
coords <- coordinates(as(columbus, "Spatial"))
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 = NULL,
  scale = 1, ...)
## S3 method for class 'listw'
print(x, zero.policy = NULL, ...)
```



**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 NULL, use global option value; if FALSE stop with error for any empty neighbour sets
scale	passed through to stem() for control of plot length

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**See Also**

[plot.nb](#)

**Examples**

```
columbus <- st_read(system.file("shapes/columbus.shp", 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)
```

---

tolerance.nb	<i>Function to construct edges based on a tolerance angle and a maximum distance</i>
--------------	--

---

**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 bottom to the top of the screen.

**Usage**

```
tolerance.nb(coords, unit.angle = "degrees", max.dist, tolerance, rot.angle,
plot.sites=FALSE)
```

**Arguments**

<code>coords</code>	A matrix or a data frame containing the X and Y coordinates of the study sites.
<code>unit.angle</code>	Character. The measurement units in which angles are defined: either "degrees" (default) or "radians".
<code>max.dist</code>	Numeric. The maximum distance of an edge linking two sites together.
<code>tolerance</code>	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.
<code>rot.angle</code>	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.
<code>plot.sites</code>	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**

[dnearest](#), [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

<code>coords</code>	matrix of point coordinates with two columns, a <code>SpatialPoints</code> object or an <code>sfc</code> points object
<code>row.names</code>	character vector of region ids to be added to the neighbours list as attribute <code>region.id</code> , default <code>seq(1, nrow(x))</code>

## 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](#), [dnearest](#), [cell2nb](#)

**Examples**

```
columbus <- st_read(system.file("shapes/columbus.shp", 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,])
```

---

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

<code>oldstyle</code>	if TRUE, first line of file contains only number of spatial units, if FALSE, uses newer GeoDa style
<code>shpfile</code>	Shapefile name taken from GAL file for this dataset
<code>ind</code>	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

- \*Topic **cluster**
  - nbcosts, 103
  - plot.skater, 111
  - prunecost, 115
  - prunemst, 116
  - skater, 125
  - ssw, 165
- \*Topic **datasets**
  - columbus, 18
  - eire, 28
  - oldcol, 106
- \*Topic **data**
  - bhcv, 13
- \*Topic **graphs**
  - mstree, 92
  - prunecost, 115
- \*Topic **hplot**
  - plot.mst, 109
  - plot.skater, 111
- \*Topic **manip**
  - Rotation, 120
- \*Topic **multivariate**
  - ssw, 165
- \*Topic **spatial**
  - aggregate.nb, 5
  - airdist, 6
  - aple, 7
  - aple.mc, 8
  - aple.plot, 10
  - autocov\_dist, 11
  - card, 14
  - cell2nb, 15
  - choynowski, 16
  - diffnb, 18
  - dnearneigh, 19
  - droplinks, 21
  - EBest, 22
  - EBI Moran.mc, 23
  - EBlocal, 26
  - edit.nb, 27
  - geary, 28
  - geary.mc, 30
  - geary.test, 31
  - globalG.test, 33
  - Graph Components, 35
  - graphneigh, 36
  - grid2nb, 40
  - include.self, 41
  - invIrM, 42
  - is.symmetric.nb, 44
  - joincount.mc, 46
  - joincount.multi, 47
  - joincount.test, 49
  - knearneigh, 51
  - knn2nb, 53
  - lag.listw, 54
  - lee, 55
  - lee.mc, 56
  - lee.test, 58
  - listw2sn, 60
  - lm.LMtests, 61
  - lm.morantest, 63
  - lm.morantest.exact, 65
  - lm.morantest.sad, 67
  - localG, 69
  - localmoran, 71
  - localmoran.exact, 73
  - localmoran.sad, 76
  - LOSH, 79
  - LOSH.cs, 81
  - LOSH.mc, 82
  - mat2listw, 84
  - moran, 85
  - moran.mc, 86
  - moran.plot, 88
  - moran.test, 90
  - mstree, 92
  - nb.set.operations, 94

- nb2blocknb, 95
- nb2INLA, 96
- nb2lines, 97
- nb2listw, 99
- nb2mat, 101
- nb2WB, 102
- nbcosts, 103
- nbdists, 104
- nblag, 105
- p.adjustSP, 108
- plot.nb, 110
- poly2nb, 112
- probmap, 113
- read.gal, 117
- read.gwt2nb, 118
- set.mcOption, 121
- set.spChkOption, 123
- sp.correlogram, 129
- sp.mantel.mc, 131
- spdep, 133
- spdep-deprecated, 133
- spweights.constants, 164
- subset.listw, 166
- subset.nb, 167
- summary.nb, 168
- tolerance.nb, 169
- tri2nb, 171
- write.nb.gal, 172
- \*Topic **tree**
  - plot.mst, 109
  - prunemst, 116
  - skater, 125
- aggregate.nb, 5
- airdist, 6
- anova.sarlm (spdep-deprecated), 133
- aple, 7, 9, 11
- aple.mc, 8, 8, 122
- aple.plot, 8, 10
- as.data.frame.localmoranex (localmoran.exact), 73
- as.data.frame.localmoransad (localmoran.sad), 76
- as.data.frame.sarlm.pred (spdep-deprecated), 133
- as.spam.listw (spdep-deprecated), 133
- as\_dgRMatrix\_listw (spdep-deprecated), 133
- as\_dsCMatrix\_I (spdep-deprecated), 133
- as\_dsCMatrix\_IrW (spdep-deprecated), 133
- as\_dsTMatrix\_listw (spdep-deprecated), 133
- autocov\_dist, 11
- bbs (columbus), 18
- bhicv, 13
- boot, 9
- bptest.sarlm (spdep-deprecated), 133
- can.be.simmed (spdep-deprecated), 133
- card, 14, 16, 20, 38, 40, 53, 112, 113
- cell2nb, 15, 96, 170, 172
- cheb\_setup (spdep-deprecated), 133
- chkIDs (set.spChkOption), 123
- Cholesky, 140
- choynowski, 16
- coef.gmsar (spdep-deprecated), 133
- coef.lagmess (spdep-deprecated), 133
- coef.sarlm (spdep-deprecated), 133
- coef.spautolm (spdep-deprecated), 133
- coef.stsls (spdep-deprecated), 133
- coerce, listw, CsparseMatrix-method (spdep-deprecated), 133
- coerce, listw, RsparseMatrix-method (spdep-deprecated), 133
- coerce, listw, symmetricMatrix-method (spdep-deprecated), 133
- col.gal.nb (columbus), 18
- COL.nb (oldcol), 106
- COL.OLD (oldcol), 106
- columbus, 18
- complement.nb (nb.set.operations), 94
- coords (columbus), 18
- create\_WX (spdep-deprecated), 133
- Deprecated, 142
- deviance.gmsar (spdep-deprecated), 133
- deviance.lagmess (spdep-deprecated), 133
- deviance.sarlm (spdep-deprecated), 133
- deviance.spautolm (spdep-deprecated), 133
- deviance.stsls (spdep-deprecated), 133
- df2sn (nb2lines), 97
- diffnb, 18
- dist, 104, 115, 125, 165
- dnearneigh, 19, 38, 52, 96, 170, 172
- do\_ldet (spdep-deprecated), 133
- dput, 103

- droplinks, 21
- EBest, 22, 25, 27, 114
- EImoran (EImoran.mc), 23
- EImoran.mc, 23, 23
- EBlocal, 23, 26, 114
- edit.nb, 27
- eigen\_pre\_setup (spdep-deprecated), 133
- eigen\_setup (spdep-deprecated), 133
- eigenw (spdep-deprecated), 133
- eire, 28
- errorsarlm, 78
- errorsarlm (spdep-deprecated), 133
- fitted.gmsar (spdep-deprecated), 133
- fitted.lagmess (spdep-deprecated), 133
- fitted.ME\_res (spdep-deprecated), 133
- fitted.sarlm (spdep-deprecated), 133
- fitted.SFResult (spdep-deprecated), 133
- fitted.spauto1m (spdep-deprecated), 133
- gabrielneigh (graphneigh), 36
- geary, 28, 31, 33
- geary.mc, 29, 30, 33, 132
- geary.test, 29–31, 31
- geos\_unary, 37
- get.ClusterOption (set.mcOption), 121
- get.coresOption (set.mcOption), 121
- get.listw\_is\_CsparseMatrix\_Option (set.spChkOption), 123
- get.mcOption (set.mcOption), 121
- get.spChkOption (set.spChkOption), 123
- get.VerboseOption (set.spChkOption), 123
- get.ZeroPolicyOption (set.spChkOption), 123
- globalG.test, 33
- GMargminImage (spdep-deprecated), 133
- GMerrorsar (spdep-deprecated), 133
- Graph Components, 35
- graph2nb (graphneigh), 36
- graphneigh, 36, 170
- grid2nb, 39
- griffith\_sone (spdep-deprecated), 133
- gstsls (spdep-deprecated), 133
- Hausman.test (spdep-deprecated), 133
- HPDinterval.lagImpact (spdep-deprecated), 133
- impacts (spdep-deprecated), 133
- include.self, 41
- influence.measures, 89
- intersect.nb, 94
- intersect.nb (nb.set.operations), 94
- intImpacts (spdep-deprecated), 133
- invIrM, 42
- invIrW (invIrM), 42
- is.symmetric.glist (is.symmetric.nb), 44
- is.symmetric.nb, 21, 44
- Jacobian\_W (spdep-deprecated), 133
- jacobianSetup (spdep-deprecated), 133
- joincount.mc, 46, 50, 132
- joincount.multi, 47, 50
- joincount.test, 47, 48, 49
- knearneigh, 20, 38, 51, 53
- knn, 52
- knn2nb, 38, 52, 53, 96, 170, 172
- l\_max (spdep-deprecated), 133
- lag.listw, 54
- lagmess (spdep-deprecated), 133
- lagsarlm (spdep-deprecated), 133
- lee, 55, 57, 59
- lee.mc, 55, 56, 59
- lee.test, 58
- lextrB (spdep-deprecated), 133
- lextrS (spdep-deprecated), 133
- lextrW (spdep-deprecated), 133
- listw2lines (nb2lines), 97
- listw2mat (nb2mat), 101
- listw2sn, 60
- listw2star (localmoran.sad), 76
- listw2U, 33, 50, 59, 91
- listw2U (lm.morantest), 63
- listw2WB (nb2WB), 102
- lm, 62, 64
- lm.LMtests, 61, 64
- lm.morantest, 63, 68, 78
- lm.morantest.exact, 65, 75
- lm.morantest.sad, 66, 67, 78
- lmSLX (spdep-deprecated), 133
- localAple (aple.plot), 10
- localG, 33, 34, 69, 73, 108
- localmoran, 71, 78, 89, 108
- localmoran.exact, 73
- localmoran.sad, 75, 76
- locator, 6



- logLik.lagmess (spdep-deprecated), 133
- logLik.sarlm (spdep-deprecated), 133
- logLik.spautolm (spdep-deprecated), 133
- LOSH, 79, 81–84
- LOSH.cs, 80, 81
- LOSH.mc, 80, 82, 82, 84
- LR.sarlm (spdep-deprecated), 133
- LR1.sarlm (spdep-deprecated), 133
- LR1.spautolm (spdep-deprecated), 133
- LU\_prepermutate\_setup (spdep-deprecated), 133
- LU\_setup (spdep-deprecated), 133
- make.sym.nb (is.symmetric.nb), 44
- mat2listw, 84
- Matrix\_J\_setup (spdep-deprecated), 133
- Matrix\_setup (spdep-deprecated), 133
- mcdet\_setup (spdep-deprecated), 133
- MCMCsamp (spdep-deprecated), 133
- ME (spdep-deprecated), 133
- mom\_calc (spdep-deprecated), 133
- mom\_calc\_int2 (spdep-deprecated), 133
- moments\_setup (spdep-deprecated), 133
- moran, 25, 85, 87, 91, 130
- moran.mc, 25, 86, 86, 91, 132
- moran.plot, 88
- moran.test, 86, 87, 90
- mrc2vi (cell2nb), 15
- mstree, 92, 109, 111, 117, 126
- n.comp.nb (Graph Components), 35
- nb.set.operations, 94
- nb2blocknb, 95
- nb2INLA, 96
- nb2lines, 97
- nb2listw, 8, 12, 43, 54, 60, 84, 85, 92, 99, 102, 104, 105, 165, 167, 168
- nb2mat, 85, 101
- nb2WB, 102
- nbcost, 166
- nbcost (nbcosts), 103
- nbcosts, 103
- nbdists, 104, 104
- nblag, 105, 130
- nblag\_cumul (nblag), 105
- nn2, 52
- old.make.sym.nb (is.symmetric.nb), 44
- oldcol, 106
- p.adjust, 62, 108, 130
- p.adjustSP, 72, 81, 83, 108
- plot.Gabriel (graphneigh), 36
- plot.lagImpact (spdep-deprecated), 133
- plot.listw (plot.nb), 110
- plot.mc.sim (sp.mantel.mc), 131
- plot.mst, 109
- plot.nb, 28, 36, 110, 169
- plot.relative (graphneigh), 36
- plot.skater, 111
- plot.spcor (sp.correlogram), 129
- poly2nb, 40, 96, 104, 112
- polys (columbus), 18
- powerWeights (spdep-deprecated), 133
- ppois, 114
- predict.sarlm (spdep-deprecated), 133
- predict.SLX (spdep-deprecated), 133
- print.gmsar (spdep-deprecated), 133
- print.jclist (joincount.test), 49
- print.jcmulti (joincount.multi), 47
- print.lagImpact (spdep-deprecated), 133
- print.lagmess (spdep-deprecated), 133
- print.listw (summary.nb), 168
- print.LMtestlist (lm.LMtests), 61
- print.localmoranex (localmoran.exact), 73
- print.localmoransad (localmoran.sad), 76
- print.ME\_res (spdep-deprecated), 133
- print.moranex (lm.morantest.exact), 65
- print.moransad (lm.morantest.sad), 67
- print.nb (summary.nb), 168
- print.sarlm (spdep-deprecated), 133
- print.SFResult (spdep-deprecated), 133
- print.spautolm (spdep-deprecated), 133
- print.spcor (sp.correlogram), 129
- print.stsls (spdep-deprecated), 133
- print.summary.gmsar (spdep-deprecated), 133
- print.summary.lagImpact (spdep-deprecated), 133
- print.summary.lagmess (spdep-deprecated), 133
- print.summary.localmoransad (localmoran.sad), 76
- print.summary.moransad (lm.morantest.sad), 67
- print.summary.sarlm (spdep-deprecated), 133

- print.summary.spautolm (spdep-deprecated), 133
- print.summary.stsls (spdep-deprecated), 133
- probmap, 17, 23, 27, 113
- prunecost, 115
- prunemst, 116, 116
- queencecell (cell2nb), 15
- read.dat2listw (read.gwt2nb), 118
- read.gal, 45, 100, 117, 119, 173
- read.geoda (read.gal), 117
- read.gwt2nb, 118
- relativeneigh (graphneigh), 36
- residuals.gmsar (spdep-deprecated), 133
- residuals.lagmess (spdep-deprecated), 133
- residuals.sarlm (spdep-deprecated), 133
- residuals.spautolm (spdep-deprecated), 133
- residuals.stsls (spdep-deprecated), 133
- rookcell (cell2nb), 15
- Rotation, 120
- sacsarlm (spdep-deprecated), 133
- SE\_classic\_setup (spdep-deprecated), 133
- SE\_interp\_setup (spdep-deprecated), 133
- SE\_whichMin\_setup (spdep-deprecated), 133
- set.ClusterOption (set.mcOption), 121
- set.coresOption (set.mcOption), 121
- set.listw\_is\_CsparseMatrix\_Option (set.spChkOption), 123
- set.mcOption, 121
- set.spChkOption, 123
- set.VerboseOption (set.spChkOption), 123
- set.ZeroPolicyOption (set.spChkOption), 123
- setdiff.nb, 94
- setdiff.nb (nb.set.operations), 94
- similar.listw (spdep-deprecated), 133
- skater, 109, 111, 125
- sn2listw, 98
- sn2listw (listw2sn), 60
- soi.graph (graphneigh), 36
- sp.correlogram, 129
- sp.mantel.mc, 29, 131
- spam\_setup (spdep-deprecated), 133
- spam\_update\_setup (spdep-deprecated), 133
- SpatialFiltering (spdep-deprecated), 133
- spautolm (spdep-deprecated), 133
- spBreg\_err (spdep-deprecated), 133
- spBreg\_lag (spdep-deprecated), 133
- spBreg\_sac (spdep-deprecated), 133
- spdep, 133
- spdep-deprecated, 133
- spNamedVec (set.spChkOption), 123
- spweights.constants, 164
- ssw, 165
- stsls (spdep-deprecated), 133
- subgraph\_eigenw (spdep-deprecated), 133
- subset.listw, 166
- subset.nb, 167, 167
- summary.gmsar (spdep-deprecated), 133
- summary.lagImpact (spdep-deprecated), 133
- summary.lagmess (spdep-deprecated), 133
- summary.listw (summary.nb), 168
- summary.LMtestlist (lm.LMtests), 61
- summary.localmoransad (localmoran.sad), 76
- summary.moransad (lm.morantest.sad), 67
- summary.nb, 15, 16, 28, 40, 41, 100, 105, 106, 110, 113, 118, 168
- summary.sarlm (spdep-deprecated), 133
- summary.spautolm (spdep-deprecated), 133
- summary.stsls (spdep-deprecated), 133
- sym.attr.nb (is.symmetric.nb), 44
- Szero (spweights.constants), 164
- tolerance.nb, 169
- tri2nb, 96, 170, 171
- trW, 140
- trW (spdep-deprecated), 133
- union.nb, 94
- union.nb (nb.set.operations), 94
- vcov.sarlm (spdep-deprecated), 133
- vi2mrc (cell2nb), 15
- Wald1.sarlm (spdep-deprecated), 133
- write.nb.gal, 172
- write.sn2dat (read.gwt2nb), 118
- write.sn2gwt (read.gwt2nb), 118