# The spdep Package

April 19, 2005

**Version** 0.3-12

**Date** 2005-04-19

**Title** Spatial dependence: weighting schemes, statistics and models

**Author** Roger Bivand <Roger.Bivand@nhh.no>, with contributions by Luc Anselin, Andrew Bernat, Marilia Carvalho, Stéphane Dray, Rein Halbersma, Nicholas Lewin-Koh, Hisaji Ono, Michael Tiefelsdorf, and Danlin Yu.

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

**Depends** R (>= 2.0.0), tripack, maptools (>= 0.4-11), SparseM (>= 0.54)

**Description** A collection of functions to create spatial weights matrix objects from polygon contiguities, from point patterns by distance and tesselations, for summarising these objects, and for permitting their use in spatial data analysis; a collection of tests for spatial autocorrelation, including global Moran's I, Geary's C, Hubert/Mantel general cross product statistic, Empirical Bayes estimates and Assunção/Reis Index, Getis/Ord G and multicoloured join count statistics, local Moran's I and Getis/Ord G, saddlepoint approximations for global and local Moran's I; and functions for estimating spatial simultaneous autoregressive (SAR) models.

**License** GPL version 2 or newer

## R topics documented:

---

`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 a `polylist` object `polys.OLD` with the boundaries of the neighbourhoods, a matrix of polygon centroids `coords.OLD`, and `COL.nb`, the neighbours list as used in Anselin (1988).

## Usage

```
data(oldcol)
```

## Format

This data frame contains the following columns:

**AREA** computed by ArcView

**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** 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, http://sal.agecon.uiuc.edu/datasets/columbus.zip.

## Source

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

---

# EBImoran.mc *Permutation test for empirical Bayes index*

---

## Description

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

## Usage

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

## 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 | 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() |

## Details

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

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

where:

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

and:

$$p_i = n_i / x_i$$

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

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

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

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

## Value

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

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Assunção RM, Reis EA 1999 A new proposal to adjust Moran's I for population density. Statistics in Medicine 18, pp. 2147–2162

### See Also

`moran`, `moran.mc`, `EBest`

### Examples

```
data(nc.sids)
EBImoran.mc(spNamedVec("SID74", nc.sids), spNamedVec("BIR74", nc.sids),
 nb2listw(ncCC89.nb, style="B", zero.policy=TRUE), nsim=999,
 zero.policy=TRUE)
sids.p <- nc.sids$SID74 / nc.sids$BIR74
names(sids.p) <- rownames(nc.sids)
moran.mc(sids.p, nb2listw(ncCC89.nb, style="B", zero.policy=TRUE),
 nsim=999, zero.policy=TRUE)
```

---

## EBest *Global Empirical Bayes estimator*

---

### Description

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

### Usage

```
EBest(n, x)
```

### Arguments

| | |
|---|---|
| `n` | a numeric vector of counts of cases |
| `x` | a numeric vector of populations at risk |

### Details

Details of the implementation are to be found in Marshall, p. 284–5, and Bailey and Gatrell p. 303–306 and exercise 8.2, pp. 328–330.

### Value

A data frame with two columns:

| | |
|---|---|
| `raw` | a numerical vector of raw (crude) rates |
| `estmm` | a numerical vector of empirical Bayes estimates |
| `a` | global method of moments phi value |
| `m` | global method of moments gamma value |

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### 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

EBlocal, probmap, EBImoran.mc

### Examples

```
data(auckland)
res <- EBest(auckland$Deaths.1977.85, 9*auckland$Under.5.1981)
attr(res, "parameters")
cols <- grey(6:2/7)
brks <- c(-Inf,2,2.5,3,3.5,Inf)
library(maptools)
plot(auckpolys, col=cols[findInterval(res$estmm*1000, brks)], forcefill=FALSE)
legend(c(70,90), c(70,95), fill=cols, legend=leglabs(brks), bty="n")
title(main="Global moment estimator of infant mortality per 1000 per year")
```

---

## EBlocal *Local Empirical Bayes estimator*

---

### Description

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

### Usage

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

### Details

Details of the implementation are to be found in Marshall, p. 286, and Bailey and Gatrell p. 307 and exercise 8.2, pp. 328–330. The example results do not fully correspond to the sources because of slightly differing neighbourhoods, but are generally close.

### Value

A data frame with two columns:

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

```
data(auckland)
res <- EBlocal(spNamedVec("Deaths.1977.85", auckland),
 9*spNamedVec("Under.5.1981", auckland), auckland.nb)
brks <- c(-Inf,2,2.5,3,3.5,Inf)
cols <- grey(6:2/7)
library(maptools)
plot(auckpolys, col=cols[findInterval(res$est*1000, brks)], forcefill=FALSE)
legend(c(70,90), c(70,95), fill=cols, legend=leglabs(brks), bty="n")
title(main="Local moment estimator of infant mortality per 1000 per year")
```

---

## `LR.sarlm` *Likelihood ratio test*

---

### Description

The `LR.sarlm()` function provides a likelihood ratio test for objects for which a `logLik()` function exists for their class, or for objects of class `logLik`. `LR1.sarlm()` and `Wald1.sarlm()` are used internally in `summary.sarlm()`, but may be accessed directly; they report the values respectively of LR and Wald tests for the absence of spatial dependence in spatial lag or error models.

### Usage

```
LR.sarlm(x, y)
logLik.sarlm(object, ...)
LR1.sarlm(object)
Wald1.sarlm(object)
```

### Arguments

| | |
|---|---|
| `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 |
| `object` | a `sarlm` object from `lagsarlm()` or `errorsarlm()` |
| `...` | other arguments to logLik() |

### Value

`LR.sarlm()` returns an object of class `htest` with:

| | |
|---|---|
| `statistic` | value of statistic |
| `parameter` | degrees of freedom |
| `p.value` | Probability value |
| `estimate` | Log likelihood values of compared models |
| `method` | "Likelihood ratio for spatial linear models" |

`logLik.sarlm()` returns an object of class `logLik` `LR1.sarlm` and `Wald1.sarlm` return objects of class `htest`

### Note

The numbers of degrees of freedom returned by `logLik.sarlm()` include nuisance parameters, that is the number of regression coefficients, plus sigma, plus spatial parameter esitmate(s).

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

`logLik.lm`, `anova.sarlm`

### Examples

```
data(columbus)
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)
```

afcon *Spatial patterns of conflict in Africa 1966-78*

## Description

The afcon data frame has 42 rows and 5 columns, for 42 African countries, exclusing then South West Africa and Spanish Equatorial Africa and Spanish Sahara. The dataset is used in Anselin (1995), and downloaded from before adaptation. The neighbour list object africa.rook.nb is the SpaceStat 'rook.GAL', but is not the list used in Anselin (1995) - paper.nb reconstructs the list used in the paper, with inserted links between Mauritania and Morocco, South Africa and Angola and Zambia, Tanzania and Zaire, and Botswana and Zambia. afxy is the coordinate matrix for the centroids of the countries.

## Usage

```
data(afcon)
```

## Format

This data frame contains the following columns:

**x** an easting in decimal degrees (taken as centroid of shapefile polygon)

**y** an northing in decimal degrees (taken as centroid of shapefile polygon)

**totcon** index of total conflict 1966-78

**name** country name

**id** country id number as in paper

## Note

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

## Source

Anselin, L. and John O'Loughlin. 1992. Geography of international conflict and cooperation: spatial dependence and regional context in Africa. In The New Geopolitics, ed. M. Ward, pp. 39-75. Philadelphia, PA: Gordon and Breach. also: Anselin, L. 1995. Local indicators of spatial association, Geographical Analysis, 27, Table 1, p. 103.

## Examples

```
data(afcon)
plot(africa.rook.nb, afxy)
plot(diffnb(paper.nb, africa.rook.nb), afxy, col="red", add=TRUE)
text(afxy, labels=attr(africa.rook.nb, "region.id"), pos=4, offset=0.4)
moran.test(spNamedVec("totcon", afcon), nb2listw(africa.rook.nb))
moran.test(spNamedVec("totcon", afcon), nb2listw(paper.nb))
geary.test(spNamedVec("totcon", afcon), nb2listw(paper.nb))
```

## 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`

## anova.sarlm *Comparison of simultaneous autoregressive models*

### Description

One of a number of tools for comparing simultaneous autoregressive models, in particular nested models. The function is based on `anova.lme()` for comparing linear mixed models, and follows that function in using the "anova" generic name.

### Usage

```
anova.sarlm(object, ...)
```

### Arguments

| | |
|---|---|
| `object` | `object` is of class `sarlm` |
| `...` | other objects of class `sarlm` or class `lm` |

### Details

If successive models have different numbers of degrees of freedom, a likelihood ratio test will be performed between them. It is important to recall that tests apply to nested models, and this function at least attempts to make sure that the response variable in the models being compared has the same name. Useless results can still be generated when incomparable models are compared, it being the responsibility of the user to check.

### Value

The function returns a data frame printed by default functions

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

`LR.sarlm`, `AIC`

### Examples

```
data(columbus)
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),
  type="mixed")
error <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
LR.sarlm(mixed, error)
anova(lag, lm.mod)
anova(lag, error, mixed)
AIC(lag, error, mixed)
```

---

## `asMatrixCsrListw` *Convert spatial weights list to CSR matrix form*

---

### Description

`asMatrixCsrListw` creates a `matrix.csr` from a `listw` object, `asMatrixCsrIrW` is a helper function used in fitting SAR models and elsewhere to make a (I - rho W) `matrix.csr` object from W; `asListwMatrixCsr` performs the reverse conversion from `matrix.csr` to `listw` object.

### Usage

```
asMatrixCsrListw(listw)
asMatrixCsrIrW(W, rho)
asMatrixCsrI(n)
asListwMatrixCsr(mcsr)
```

### Arguments

| | |
|---|---|
| listw | a listw object created for example by nb2listw |
| W | a matrix.csr object created using asMatrixCsrListw from a listw object |
| rho | spatial regression coefficient |
| n | length of diagonal for identity matrix |
| mcsr | a matrix.csr object |

### Value

asMatrixCsrListw and asMatrixCsrIrW return matrix.csr objects; asListwMatrixCsr returns a listw object

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

as.matrix.csr

### Examples

```
data(oldcol)
COL.W <- nb2listw(COL.nb, style="W")
COL.W
image(asMatrixCsrListw(COL.W))
```

---

## auckland *Marshall's infant mortality in Auckland dataset*

---

### Description

The auckland data frame has 167 rows (census area units — CAU) and 4 columns. The dataset also includes the "nb" object auckland.nb of neighbour relations based on contiguity, and the "polylist" object auckpolys of polygon boundaries for the CAU. The auckland data frame includes the following columns:

### Usage

```
data(auckland)
```

### Format

This data frame contains the following columns:

**Easting** a numeric vector of x coordinates in an unknown spatial reference system

**Northing** a numeric vector of y coordinates in an unknown spatial reference system

**Deaths.1977.85** a numeric vector of counts of infant (under 5 years of age) deaths in Auckland, 1977-1985

**Under.5.1981** a numeric vector of population under 5 years of age at the 1981 Census

### Details

The contiguous neighbours object does not completely replicate results in the sources, and was reconstructed from `auckpolys`; examination of figures in the sources suggests that there are differences in detail, although probably not in substance.

### Source

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 — INFOMAP data set used with permission.

---

## `baltimore` *House sales prices, Baltimore, MD 1978*

---

### Description

House sales price and characteristics for a spatial hedonic regression, Baltimore, MD 1978. X,Y on Maryland grid, projection type unknown.

### Usage

```
data(baltimore)
```

### Format

A data frame with 211 observations on the following 17 variables.

**STATION** a numeric vector

**PRICE** a numeric vector

**NROOM** a numeric vector

**DWELL** a numeric vector

**NBATH** a numeric vector

**PATIO** a numeric vector

**FIREPL** a numeric vector

**AC** a numeric vector

**BMENT** a numeric vector

**NSTOR** a numeric vector

**GAR** a numeric vector

**AGE** a numeric vector

**CITCOU** a numeric vector

**LOTSZ** a numeric vector

**SQFT** a numeric vector

**X** a numeric vector

**Y** a numeric vector

### Source

Prepared by Luc Anselin. Original data made available by Robin Dubin, Weatherhead School of Management, Case Western Research University, Cleveland, OH. http://sal.agecon.uiuc.edu/datasets/baltimore.zip

### References

Dubin, Robin A. (1992). Spatial autocorrelation and neighborhood quality. Regional Science and Urban Economics 22(3), 433-452.

### Examples

```
data(baltimore)
## maybe str(baltimore) ; plot(baltimore) ...
```

---

## boston *Corrected Boston Housing Data*

---

### Description

The `boston.c` data frame has 506 rows and 20 columns. It contains the Harrison and Rubinfeld (1978) data corrected for a few minor errors and augmented with the latitude and longitude of the observations. Gilley and Pace also point out that MEDV is censored, in that median values at or over USD 50,000 are set to USD 50,000. The original data set without the corrections is also included in package `mlbench` as `BostonHousing`. In addition, a matrix of tract point coordinates projected to UTM zone 19 is included as `boston.utm`, and a sphere of influence neighbours list as `boston.soi`.

### Usage

```
data(boston)
```

### Format

This data frame contains the following columns:

**TOWN** a factor with levels given by town names

**TOWNNO** a numeric vector corresponding to TOWN

**TRACT** a numeric vector of tract ID numbers

**LON** a numeric vector of tract point longitudes in decimal degrees

**LAT** a numeric vector of tract point latitudes in decimal degrees

**MEDV** a numeric vector of median values of owner-occupied housing in USD 1000

**CMEDV** a numeric vector of corrected median values of owner-occupied housing in USD 1000

**CRIM** a numeric vector of per capita crime

**ZN** a numeric vector of proportions of residential land zoned for lots over 25000 sq. ft per town (constant for all Boston tracts)

**INDUS** a numeric vector of proportions of non-retail business acres per town (constant for all Boston tracts)

**CHAS** a factor with levels 1 if tract borders Charles River; 0 otherwise

**NOX** a numeric vector of nitric oxides concentration (parts per 10 million) per town

**RM** a numeric vector of average numbers of rooms per dwelling

**AGE** a numeric vector of proportions of owner-occupied units built prior to 1940

**DIS** a numeric vector of weighted distances to five Boston employment centres

**RAD** a numeric vector of an index of accessibility to radial highways per town (constant for all Boston tracts)

**TAX** a numeric vector full-value property-tax rate per USD 10,000 per town (constant for all Boston tracts)

**PTRATIO** a numeric vector of pupil-teacher ratios per town (constant for all Boston tracts)

**B** a numeric vector of `1000*(Bk - 0.63)^2` where Bk is the proportion of blacks

**LSTAT** a numeric vector of percentage values of lower status population

## Source

http://lib.stat.cmu.edu/datasets/boston_corrected.txt

## References

Harrison, David, and Daniel L. Rubinfeld, Hedonic Housing Prices and the Demand for Clean Air, *Journal of Environmental Economics and Management*, Volume 5, (1978), 81-102. Original data.

Gilley, O.W., and R. Kelley Pace, On the Harrison and Rubinfeld Data, *Journal of Environmental Economics and Management*, 31 (1996), 403-405. Provided corrections and examined censoring.

Pace, R. Kelley, and O.W. Gilley, Using the Spatial Configuration of the Data to Improve Estimation, *Journal of the Real Estate Finance and Economics*, 14 (1997), 333-340.

## Examples

```
data(boston)
hr0 <- lm(log(MEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) +
 AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT), data=boston.c)
summary(hr0)
logLik(hr0)
gp0 <- lm(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)
summary(gp0)
logLik(gp0)
lm.morantest(hr0, nb2listw(boston.soi))
gp1 <- errorsarlm(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), method="SparseM",
 tol.opt = .Machine$double.eps^(1/4))
summary(gp1)
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, nb2listw(boston.soi), method="SparseM")
summary(gp2)
```

`bptest.sarlm` *Breusch-Pagan test for spatial models*

## Description

Performs the Breusch-Pagan test for heteroskedasticity on the least squares fit of the spatial models taking the spatial coefficients rho or lambda into account. This function is a copy of the `bptest` function in package "lmtest", modified to use objects returned by spatial simultaneous autoregressive models.

## Usage

```
bptest.sarlm(object, studentize = TRUE)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `"sarlm"` from `errorsarlm()` or `lagsarlm()`. |
| `studentize` | logical. If set to `TRUE` Koenker's studentized version of the test statistic will be used. |

## Details

Asymptotically this corresponds to the test given by Anselin (1988), but is not exactly the same. The studentized version is more conservative and perhaps to be prefered. The residuals, and for spatial error models the RHS variables, are adjusted for the spatial coefficient, as suggested bt Luc Anselin (personal communication).

## Value

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

| | |
|---|---|
| `statistic` | the value of the test statistic. |
| `p.value` | the p-value of the test. |
| `parameter` | degrees of freedom. |
| `method` | a character string indicating what type of test was performed. |

## Author(s)

Torsten Hothorn ⟨Torsten.Hothorn@rzmail.uni-erlangen.de⟩ and Achim Zeileis ⟨zeileis@ci.tuwien.ac.at⟩, modified by Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

T.S. Breusch & A.R. Pagan (1979), A Simple Test for Heteroscedasticity and Random Coefficient Variation. *Econometrica* **47**, 1287–1294

W. Krämer & H. Sonnberger (1986), *The Linear Regression Model under Test*. Heidelberg: Physica.

L. Anselin (1988) *Spatial econometrics: methods and models.* Dordrecht: Kluwer, pp. 121–122.

## See Also

`errorsarlm`, `lagsarlm`

### Examples

```
data(columbus)
error.col <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus,
 nb2listw(col.gal.nb))
bptest.sarlm(error.col)
bptest.sarlm(error.col, studentize=FALSE)
```

---

## 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 |

### Value

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

summary.nb

### Examples

```
data(columbus)
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)
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 |
| `rowcol` | matrix with two columns of row, column indices |
| `i` | vector of vector indices corresponding to rowcol |
| `rmin` | lowest row index |
| `cmin` | lowset column index |

## 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

`summary.nb`

## 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)
```

`choynowski` *Choynowski probability map values*

## Description

Calculates Choynowski probability map values.

## Usage

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

## Arguments

| | |
|---|---|
| `n` | a numeric vector of counts of cases |
| `x` | a numeric vector of populations at risk |
| `row.names` | row names passed through to output data frame |
| `tol` | accumulate values for observed counts >= expected until value less than tol |

## Value

A data frame with columns:

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

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

## See Also

`probmap`

## Examples

```
data(auckland)
res <- choynowski(auckland$Deaths.1977.85, 9*auckland$Under.5.1981)
res1 <- probmap(auckland$Deaths.1977.85, 9*auckland$Under.5.1981)
table(abs(res$pmap - res1$pmap) < 0.00001, res$type)
plot(auckpolys, forcefill=FALSE)
lt005 <- (res$pmap < 0.05) & (res$type)
ge005 <- (res$pmap < 0.05) & (!res$type)
plot(subset(auckpolys, lt005), add=TRUE, col=grey(2/7), forcefill=FALSE)
```

```
plot(subset(auckpolys, ge005), add=TRUE, col=grey(5/7), forcefill=FALSE)
legend(c(70,90), c(70,95), fill=grey(c(2,5)/7),
 legend=c("low", "high"), bty="n")
```

---

`columbus` *Columbus OH spatial analysis data set*

---

## Description

The `columbus` data frame has 49 rows and 22 columns. Unit of analysis: 49 neighbourhoods in Columbus, OH, 1980 data. In addition the data set includes a `polylist` object `polys` with the boundaries of the neighbourhoods, a matrix of polygon centroids `coords`, and `col.gal.nb`, the neighbours list from an original GAL-format file. The matrix `bbs` is DEPRECATED, but retained for other packages using this data set.

## Usage

```
data(columbus)
```

## Format

This data frame contains the following columns:

**AREA** computed by ArcView

**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** 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 `columbus` and the `region.id` attribute of `polys` 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, http://sal.agecon.uiuc.edu/datasets/columbus.zip.

### Source

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

### Examples

```
data(columbus)
summary(columbus)
```

---

## `Graph Components` *Depth First Search on Neighbor Lists*

---

### Description

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

### Usage

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

### Arguments

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

### Value

A list of:

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

### Author(s)

Nicholas Lewin-Koh ⟨kohnicho@comp.nus.edu.sg⟩

### See Also

`plot.nb`

### Examples

```
data(columbus)
plot(col.gal.nb, coords, col="grey")
col2 <- droplinks(col.gal.nb, 21)
plot(col2, coords, add=TRUE)
res <- n.comp.nb(col2)
table(res$comp.id)
points(coords, col=res$comp.id, pch=16)
```

---

## 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=TRUE)
```

### Arguments

| | |
|---|---|
| `x` | an object of class `nb` |
| `y` | an object of class `nb` |
| `verbose` | 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

```
data(columbus)
knn1 <- knearneigh(coords, 1)
knn2 <- knearneigh(coords, 2)
nb1 <- knn2nb(knn1, row.names=rownames(columbus))
nb2 <- knn2nb(knn2, row.names=rownames(columbus))
diffs <- diffnb(nb2, nb1)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(nb1, coords, add=TRUE)
plot(diffs, coords, add=TRUE, col="red", lty=2)
title(main="Plot of first (black) and second (red)\nnearest neighbours")
```

`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 lonlat = TRUE, by Great Circle distance in kilometers.

## Usage

```
dnearneigh(x, d1, d2, row.names = NULL, lonlat = FALSE)
```

## Arguments

| | |
|---|---|
| `x` | matrix of point coordinates |
| `d1` | lower distance bound |
| `d2` | upper distance bound |
| `row.names` | character vector of region ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))` |
| `lonlat` | TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers |

## Value

The function returns a list of integer vectors giving the region id numbers for neighbours satisfying the distance criteria.

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## See Also

`knearneigh`

## Examples

```
data(columbus)
k1 <- knn2nb(knearneigh(coords))
all.linked <- max(unlist(nbdists(k1, coords)))
col.nb.0.all <- dnearneigh(coords, 0, all.linked, row.names=rownames(columbus))
summary(col.nb.0.all, coords)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(col.nb.0.all, coords, add=TRUE)
title(main=paste("Distance based neighbours 0-",  format(all.linked),
 " distance units", sep=""))
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, lonlat=FALSE))
all.linked <- max(unlist(nbdists(llk1, xy, lonlat=FALSE)))
ll.nb <- dnearneigh(xy, 0, all.linked, lonlat=FALSE)
summary(ll.nb, xy, lonlat=TRUE, scale=0.5)
gck1 <- knn2nb(knearneigh(xy, k=1, lonlat=TRUE))
all.linked <- max(unlist(nbdists(gck1, xy, lonlat=TRUE)))
gc.nb <- dnearneigh(xy, 0, all.linked, lonlat=TRUE)
summary(gc.nb, xy, lonlat=TRUE, scale=0.5)
plot(ll.nb, xy)
plot(diffnb(ll.nb, gc.nb), xy, add=TRUE, col="red", lty=2)
title(main="Differences between Euclidean and Great Circle neighbours")
```

---

`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)
```

---

## 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`.

### Usage

```
edit.nb(name, coords, polys=NULL, ...)
```

### Arguments

| | |
|---|---|
| `name` | an object of class `nb` |
| `coords` | matrix of region point coordinates |
| `polys` | if polygon boundaries supplied, will be used as background |
| `...` | further arguments passed to or from other methods |

### 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`

---

## `eigenw` *Spatial weights matrix eigenvalues*

---

### Description

The function returns a numeric vector of eigenvalues of the weights matrix generated from the spatial weights object `listw`. The eigenvalues are used to speed the computation of the Jacobian in spatial SAR model estimation:

$$\log(\det[I - \rho W]) = \log \prod_{i=1}^{n} (1 - \rho \lambda_i)$$

where $W$ is the n by n spatial weights matrix, and $\lambda_i$ are the eigenvalues of $W$.

### Usage

```
eigenw(listw, quiet=TRUE)
```

### Arguments

| | |
|---|---|
| `listw` | a `listw` object created for example by `nb2listw` |
| `quiet` | set to FALSE for short summary |

### Value

a numeric vector of eigenvalues of the weights matrix generated from the spatial weights object `listw`.

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 155; Ord, J. K. 1975 Estimation methods for models of spatial interaction, Journal of the American Statistical Association, 70, 120-126.

### See Also

`eigen`,

### Examples

```
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)
```

---

## eire *Eire data sets*

---

### Description

The `eire.df` data frame has 26 rows and 9 columns. In addition, polygons of the 26 counties are provided as a multipart polylist in eire.polys.utm (coordinates in km, projection UTM zone 30). Their centroids are in eire.coords.utm. The original Cliff and Ord binary contiguities are in eire.nb.

### Usage

```
data(eire)
```

### Format

This data frame contains the following columns:

**A** Percentage of sample with blood group A

**towns** Towns/unit area

**pale** Beyond the Pale 0, within the Pale 1

**size** number of blood type samples

**ROADACC** arterial road network accessibility in 1961

**OWNCONS** percentage in value terms of gross agricultural output of each county consumed by itself

**POPCHG** 1961 population as percentage of 1926

**RETSALE** value of retail sales č000

**INCOME** total personal income č000

### Source

Upton and Fingleton 1985, - Bailey and Gatrell 1995, ch. 1 for blood group data, Cliff and Ord (1973), p. 107 for remaining variables (also after O'Sullivan, 1968). Polygon borders and Irish data sourced from Michael Tiefelsdorf's SPSS Saddlepoint bundle: `http://geog-www.sbs.ohio-state.edu/faculty/tiefelsdorf/GeoStat.htm`.

**Examples**

```
data(eire)
summary(eire.df$A)
brks <- round(fivenum(eire.df$A), digits=2)
cols <- rev(heat.colors(4))
library(maptools)
plot(eire.polys.utm,
 col=cols[findInterval(eire.df$A, brks)], forcefill=FALSE)
title(main="Percentage with blood group A in Eire")
legend(x=c(-50, 70), y=c(6120, 6050), leglabs(brks), fill=cols, bty="n")
plot(eire.polys.utm, forcefill=FALSE)
plot(eire.nb, eire.coords.utm, add=TRUE)
lA <- lag.listw(nb2listw(eire.nb), eire.df$A)
summary(lA)
moran.test(spNamedVec("A", eire.df), nb2listw(eire.nb))
geary.test(spNamedVec("A", eire.df), nb2listw(eire.nb))
cor(lA, eire.df$A)
moran.plot(spNamedVec("A", eire.df), nb2listw(eire.nb),
 labels=rownames(eire.df))
A.lm <- lm(A ~ towns + pale, data=eire.df)
summary(A.lm)
res <- residuals(A.lm)
brks <- c(min(res),-2,-1,0,1,2,max(res))
cols <- rev(cm.colors(6))
plot(eire.polys.utm, col=cols[findInterval(res, brks)], forcefill=FALSE)
title(main="Regression residuals")
legend(x=c(-50, 70), y=c(6120, 6050), legend=leglabs(brks), fill=cols,
  bty="n")
lm.morantest(A.lm, nb2listw(eire.nb))
lm.morantest.sad(A.lm, nb2listw(eire.nb))
lm.LMtests(A.lm, nb2listw(eire.nb), test="LMerr")
brks <- round(fivenum(eire.df$OWNCONS), digits=2)
cols <- grey(4:1/5)
plot(eire.polys.utm,
 col=cols[findInterval(eire.df$OWNCONS, brks)], forcefill=FALSE)
title(main="Percentage own consumption of agricultural produce")
legend(x=c(-50, 70), y=c(6120, 6050), legend=leglabs(brks),
  fill=cols, bty="n")
moran.plot(spNamedVec("OWNCONS", eire.df), nb2listw(eire.nb))
moran.test(spNamedVec("OWNCONS", eire.df), nb2listw(eire.nb))
e.lm <- lm(OWNCONS ~ ROADACC, data=eire.df)
res <- residuals(e.lm)
brks <- c(min(res),-2,-1,0,1,2,max(res))
cols <- rev(cm.colors(6))
plot(eire.polys.utm, col=cols[findInterval(res, brks)], forcefill=FALSE)
title(main="Regression residuals")
legend(x=c(-50, 70), y=c(6120, 6050), legend=leglabs(brks), fill=cm.colors(6),
  bty="n")
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
lm.LMtests(e.lm, nb2listw(eire.nb), test="LMerr")
print(localmoran.sad(e.lm, eire.nb, select=1:nrow(eire.df)))
```

---

`errorsarlm` *Spatial simultaneous autoregressive error model estimation*

---

## Description

Maximum likelihood estimation of spatial simultaneous autoregressive error models of the form:

$$y = X\beta + u, u = \lambda W u + \varepsilon$$

where $\lambda$ is found by `optimize()`, or by `optim()` using method "L-BFGS-B" if argument optim=TRUE, first, and $\beta$ and other parameters by generalized least squares subsequently.

## Usage

```
errorsarlm(formula, data=list(), listw, na.action=na.fail, method="eigen",
  quiet=TRUE, zero.policy=FALSE, interval = c(-1, 0.999), tol.solve=1.0e-10,
  tol.opt=.Machine$double.eps^0.5, control, optim=FALSE)
```

## Arguments

| | |
|---|---|
| `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. |
| `method` | "eigen" (default) - the Jacobian is computed as the product of (1 - rho*eigenvalue) using `eigenw`, and "SparseM" for strictly symmetric weights lists of styles "B", "C" and "U", or made symmetric by similarity (Ord, 1975, Appendix C) if possible for styles "W" and "S", using code from the SparseM package to calculate the determinant. |
| `quiet` | default=TRUE; if FALSE, reports function values during optimization. |
| `zero.policy` | if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `errorsarlm()` to terminate with an error |
| `interval` | search interval for autoregressive parameter when not using method="eigen"; default is c(-1,1) |
| `tol.solve` | the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to `solve()` (default=1.0e-10). This may be used if necessary to extract coefficient standard errors (for instance lowering to 1e-12), but 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 |
| `tol.opt` | the desired accuracy of the optimization - passed to `optim()` (default=square root of double precision machine tolerance, a larger root may be used if the warning: ERROR: ABNORMAL_TERMINATION_IN_LNSRCH is seen, see help(boston) for an example) |
| `control` | A list of control parameters passed to `optim`, se details in `optim` |
| `optim` | If TRUE use experimental `optim` branch and control argument |

## Details

The asymptotic standard error of $\lambda$ is only computed when method=eigen, because the full matrix operations involved would be costly for large n typically associated with the choice of method="SparseM". The same applies to the coefficient covariance matrix. Taken as the asymptotic matrix from the literature, it is typically badly scaled, being block-diagonal, and with the elements involving lambda being very small, while other parts of the matrix can be very large (often many orders of magnitude in difference). It often happens that the `tol.solve` argument needs to be set to a smaller value than the default, or the RHS variables can be centred or reduced in range.

## Value

A list object of class `sarlm`

| | |
|---|---|
| `type` | "error" |
| `lambda` | simultaneous autoregressive error coefficient |
| `coefficients` | GLS coefficient estimates |
| `rest.se` | GLS coefficient standard errors (are equal to asymptotic standard errors) |
| `LL` | log likelihood value at computed optimum |
| `s2` | GLS residual variance |
| `SSE` | sum of squared GLS errors |
| `parameters` | number of parameters estimated |
| `lm.model` | the `lm` object returned when estimating for $\lambda = 0$ |
| `method` | the method used to calculate the Jacobian |
| `call` | the call used to create this object |
| `residuals` | GLS residuals |
| `lm.target` | the `lm` object returned for the GLS fit |
| `fitted.values` | Difference between residuals and response variable |
| `ase` | TRUE if method=eigen |
| `formula` | model formula |
| `se.fit` | Not used yet |
| `lambda.se` | if ase=TRUE, the asymptotic standard error of $\lambda$ |
| `LMtest` | NULL for this model |
| `zero.policy` | zero.policy for this model |
| `na.action` | (possibly) named vector of excluded or omitted observations if non-default na.action argument used |

The internal sar.error.* functions return the value of the log likelihood function at $\lambda$.

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion; Ord, J. K. 1975 Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, 70, 120-126; Anselin, L. 1988 *Spatial econometrics: methods and models.* (Dordrecht: Kluwer); Anselin, L. 1995 SpaceStat, a software program for the analysis of spatial data, version 1.80. Regional Research Institute, West Virginia University, Morgantown, WV (www.spacestat.com); Anselin L, Bera AK (1998) Spatial dependence in linear regression models with an introduction to spatial econometrics. In: Ullah A, Giles DEA (eds) Handbook of applied economic statistics. Marcel Dekker, New York, pp. 237-289.

## See Also

lm, lagsarlm, eigenw, asMatrixCsrListw, similar.listw, predict.sarlm, residuals.sarlm

## Examples

```
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", quiet=FALSE)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", quiet=FALSE)
COL.errB.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="B"), method="eigen", quiet=FALSE)
COL.errW.SM <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="SparseM", quiet=FALSE)
summary(COL.errW.eig, correlation=TRUE)
summary(COL.errB.eig, correlation=TRUE)
summary(COL.errW.SM, correlation=TRUE)
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)
```

---

# geary *Compute Geary's C*

---

## Description

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

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

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

## Usage

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

### Arguments

| | |
|---|---|
| `x` | a numeric vector the same length as the neighbours list in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `n` | number of zones |
| `n1` | n - 1 |
| `S0` | global sum of weights |
| `zero.policy` | 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=FALSE, alternative="less", spChk=NULL)
```

### Arguments

| | |
|---|---|
| x | a numeric vector the same length as the neighbours list in listw |
| listw | a listw object created for example by nb2listw |
| nsim | number of permutations |
| zero.policy | 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 "less" (default), or "greater". |
| spChk | should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption() |

### Value

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

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

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

### See Also

geary, geary.test

### Examples

```
data(oldcol)
sim1 <- geary.mc(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="W"),
 nsim=99, alternative="less")
sim1
mean(sim1$res)
var(sim1$res)
summary(sim1$res)
colold.lags <- nblag(COL.nb, 3)
sim2 <- geary.mc(spNamedVec("CRIME", COL.OLD), nb2listw(colold.lags[[2]],
 style="W"), nsim=99)
sim2
summary(sim2$res)
sim3 <- geary.mc(spNamedVec("CRIME", COL.OLD), 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=FALSE,
    alternative="less", spChk=NULL)
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector the same length as the neighbours list in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `randomisation` | variance of I calculated under the assumption of randomisation, if FALSE normality |
| `zero.policy` | 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 "less" (default), "greater" 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()` |

## Value

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

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

## Note

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 21.

### See Also

geary, geary.mc, listw2U

### Examples

```
data(oldcol)
geary.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="W"))
geary.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
colold.lags <- nblag(COL.nb, 3)
geary.test(spNamedVec("CRIME", COL.OLD), nb2listw(colold.lags[[2]],
 style="W"))
geary.test(spNamedVec("CRIME", COL.OLD), nb2listw(colold.lags[[3]],
 style="W"), alternative="greater")
print(is.symmetric.nb(COL.nb))
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
geary.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.k4.nb, style="W"))
geary.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.k4.nb, style="W"),
 randomisation=FALSE)
cat("Note non-symmetric weights matrix - use listw2U()\n")
geary.test(spNamedVec("CRIME", COL.OLD), listw2U(nb2listw(COL.k4.nb,
 style="W")))
geary.test(spNamedVec("CRIME", COL.OLD), listw2U(nb2listw(COL.k4.nb,
 style="W")), randomisation=FALSE)
```

---

## getisord *Getis-Ord remote sensing example data*

---

### Description

The xyz data frame has 256 rows and 3 columns. Vectors x and y are of length 16 and give the centres of the grid rows and columns, 30m apart. The data start from the bootom left, Getis and Ord start from the top left - so their 136th grid cell is our 120th.

### Usage

```
data(getisord)
```

### Format

This data frame contains the following columns:

**x** grid eastings

**y** grid northings

**val** remote sensing values

### Source

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), 266.

### Examples

```
data(getisord)
image(x, y, t(matrix(xyz$val, nrow=16, ncol=16, byrow=TRUE)), asp=1)
text(xyz$x, xyz$y, xyz$val, cex=0.7)
polygon(c(195,225,225,195), c(195,195,225,225), lwd=2)
title(main="Getis-Ord 1996 remote sensing data")
```

---

## globalG.test *test for spatial autocorrelation*

---

### Description

.

### Usage

```
globalG.test(x, listw, zero.policy=FALSE, alternative="greater", spChk=NULL)
```

### 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 | 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()` |

### 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

localG

### Examples

```
data(nc.sids)
sidsrate79 <- (1000*nc.sids$SID79)/nc.sids$BIR79
names(sidsrate79) <- rownames(nc.sids)
dists <- c(10, 20, 30, 33, 40, 50, 60, 70, 80, 90, 100)
ndists <- length(dists)
ZG <- numeric(length=ndists)
milesxy <- cbind(nc.sids$east, nc.sids$north)
for (i in 1:ndists) {
  thisnb <- dnearneigh(milesxy, 0, dists[i], row.names=rownames(nc.sids))
  thislw <- nb2listw(thisnb, style="B", zero.policy=TRUE)
  ZG[i] <- globalG.test(sidsrate79, thislw, zero.policy=TRUE)$statistic
}
cbind(dists, ZG)
```

---

## graphneigh *Graph based spatial weights*

---

### Description

Functions return a graph object containing a list with the vertex coordinates and the to and from indices defining the edges. The helper function `graph2nb` converts a graph object into a neighbour list. The plot functions plot the graph objects.

### Usage

```
gabrielneigh(coords)
relativeneigh(coords)
soi.graph(tri.nb, coords)
graph2nb(gob, row.names=NULL,sym=FALSE)
plot.Gabriel(x, show.points=FALSE, add=FALSE, linecol=par(col), ...)
plot.relative(x, show.points=FALSE, add=FALSE, linecol=par(col),...)
```

### Arguments

| | |
|---|---|
| `coords` | matrix of region point coordinates |
| `tri.nb` | a neighbor list created from tri2nb |
| `gob` | a graph object created from any of the graph funtions |
| `row.names` | character vector of region ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))` |
| `sym` | a logical argument indicating whether or not neighbors should be symetric (if i->j then j->i) |

| | |
|---|---|
| x | object to be plotted |
| show.points | (logical) add points to plot |
| add | (logical) add to existing plot |
| linecol | edge plotting colour |
| ... | further graphical parameters as in par(..) |

**Details**

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

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

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

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

where x,y,z and S are as before. The sphere of influence graph is defined for a finite point set S, let $r_x$ be the distance from point x to its nearest neighbor in S, and $C_x$ is the circle centered on x. Then x and y are SOI neigbors iff $C_x$ and $C_y$ intersect in at least 2 places.

**Value**

A list of class Graph withte following elements

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

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

**Author(s)**

Nicholas Lewin-Koh ⟨kohnicho@comp.nus.edu.sg⟩

**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

### Examples

```
data(columbus)
par(mfrow=c(2,2))
col.tri.nb<-tri2nb(coords)
col.gab.nb<-graph2nb(gabrielneigh(coords), sym=TRUE)
col.rel.nb<- graph2nb(relativeneigh(coords), sym=TRUE)
col.soi.nb<- graph2nb(soi.graph(col.tri.nb,coords), sym=TRUE)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(col.tri.nb,coords,add=TRUE)
title(main="Delaunay Triangulation")
plot(polys, border="grey", forcefill=FALSE)
plot(col.gab.nb, coords, add=TRUE)
title(main="Gabriel Graph")
plot(polys, border="grey", forcefill=FALSE)
plot(col.rel.nb, coords, add=TRUE)
title(main="Relative Neighbor Graph")
plot(polys, border="grey", forcefill=FALSE)
plot(col.soi.nb, coords, add=TRUE)
title(main="Sphere of Influence Graph")
par(mfrow=c(1,1))
```

---

## hopkins *Hopkins burnt savanna herb remains*

---

### Description

A 20m square is divided into 40 by 40 0.5m quadrats. Observations are in tens of grams of herb remains, 0 being from 0g to less than 10g, and so on. Analysis was mostly conducted using the interior 32 by 32 grid.

### Usage

```
data(hopkins)
```

### Format

The format is: num [1:40, 1:40] 0 0 0 0 0 0 0 0 0 0 1 ...

### Source

Upton, G., Fingleton, B. 1985 Spatial data analysis by example: point pattern and quatitative data, Wiley, pp. 38–39.

### References

Hopkins, B., 1965 Observations on savanna burning in the Olokemeji Forest Reserve, Nigeria. Journal of Applied Ecology, 2, 367–381.

### Examples

```
data(hopkins)
image(1:32, 1:32, hopkins[5:36,36:5], breaks=c(-0.5, 3.5, 20),
 col=c("white", "black"))
box()
```

---

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

---

### Description

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

### 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

```
data(columbus)
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, a chosen coding scheme style, and optionally a list of general weights corresponding to neighbours.

## Usage

```
invIrM(neighbours, rho, glist=NULL, style="W")
invIrW(listw, rho)
```

## 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 |
| `listw` | a `listw` object from for example `nb2listw` |

## Details

The function generates the full weights matrix V, checks that rho lies in its feasible range between 1/min(eigen(V)) and 1/max(eigen(V)), and returns the nxn inverted matrix

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

## Value

An nxn matrix with a "call" attribute.

## 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.

## See Also

`nb2listw`

### Examples

```
nb7rt <- cell2nb(7, 7, torus=TRUE)
x <- matrix(rnorm(500*length(nb7rt)), nrow=length(nb7rt))
res0 <- apply(invIrM(nb7rt, rho=0.0) %*% x, 2, function(x) var(x)/length(x))
res2 <- apply(invIrM(nb7rt, rho=0.2) %*% x, 2, function(x) var(x)/length(x))
res4 <- apply(invIrM(nb7rt, rho=0.4) %*% x, 2, function(x) var(x)/length(x))
res6 <- apply(invIrM(nb7rt, rho=0.6) %*% x, 2, function(x) var(x)/length(x))
res8 <- apply(invIrM(nb7rt, rho=0.8) %*% x, 2, function(x) var(x)/length(x))
res9 <- apply(invIrM(nb7rt, rho=0.9) %*% 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=
```

---

## 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=NU
```

### Arguments

| | |
|---|---|
| fx | a factor of the same length as the neighbours and weights objects in listw |
| listw | a listw object created for example by nb2listw |
| nsim | number of permutations |
| zero.policy | 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() |

### Value

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

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

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

### See Also

`joincount.test`

### Examples

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

### Arguments

| | |
|---|---|
| fx | a factor of the same length as the neighbours and weights objects in listw |
| listw | a listw object created for example by nb2listw |
| zero.policy | 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() |
| x | object to be printed |
| ... | arguments to be passed through for printing |

### Value

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

### Note

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 20; Upton, G., Fingleton, B. 1985 Spatial data analysis by example: point pattern and quatitative 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"))
data(hopkins)
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")
```

---

`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 `geary.mc` permutations.

## Usage

```
joincount.test(fx, listw, zero.policy=FALSE, alternative="greater",
 spChk=NULL)
print.jclist(x, ...)
```

## Arguments

| | |
|---|---|
| `fx` | a factor of the same length as the neighbours and weights objects in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `zero.policy` | 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()` |
| `x` | object to be printed |
| `...` | arguments to be passed through for printing |

## Value

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

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

### Note

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 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="C"))
joincount.test(HICRIME, nb2listw(COL.nb, style="S"))
joincount.test(HICRIME, nb2listw(COL.nb, style="W"))
by(card(COL.nb), HICRIME, summary)
print(is.symmetric.nb(COL.nb))
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
joincount.test(HICRIME, nb2listw(COL.k4.nb, style="B"))
cat("Note non-symmetric weights matrix - use listw2U()\n")
joincount.test(HICRIME, listw2U(nb2listw(COL.k4.nb, style="B")))
```

---

## knearneigh *K nearest neighbours for spatial weights*

---

### Description

The function returns a matrix with the indices of regions belonging to the set of the k nearest neighbours of each other. If lonlat = TRUE, Great Circle distances are used.

### Usage

```
knearneigh(x, k=1, lonlat = FALSE)
```

### Arguments

| | |
|---|---|
| `x` | matrix of region point coordinates |
| `k` | number of nearest neighbours to be returned |
| `lonlat` | TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers |

### Details

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

### Value

A list of class `knn`

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

`knn`, `dnearneigh`, `knn2nb`

### Examples

```
data(columbus)
col.knn <- knearneigh(coords, k=4)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
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, lonlat=FALSE))
gck4.nb <- knn2nb(knearneigh(xy, k=4, lonlat=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, lonlat=TRUE)
summary(gck4.nb, xy, lonlat=TRUE)
```

`knn2nb` *Neighbours list from knn object*

## Description

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

## Usage

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

## Arguments

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

## Value

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## See Also

`knearneigh`

## Examples

```
data(columbus)
col.knn <- knearneigh(coords, k=4)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
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

```
lag.listw(x, var, zero.policy=FALSE, 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` | 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 var are passed on to the foreign function. If 'FALSE', the presence of 'NA' or 'NaN' or 'Inf' values is regarded as an error. |
| `...` | 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")
```

`lagsarlm` *Spatial simultaneous autoregressive lag model estimation*

## Description

Maximum likelihood estimation of spatial simultaneous autoregressive lag and mixed models of the form:

$$y = \rho W y + X\beta + \varepsilon$$

where $\rho$ is found by `optimize()`, or `optim()` using method "L-BFGS-B" if argument optim=TRUE, first, and $\beta$ and other parameters by generalized least squares subsequently. In the mixed model, the spatially lagged independent variables are added to X.

## Usage

```
lagsarlm(formula, data=list(), listw, na.action=na.fail,
  type="lag", method="eigen", quiet=TRUE,
  zero.policy=FALSE, interval = c(-1, 0.999), tol.solve=1.0e-10,
  tol.opt=.Machine$double.eps^0.5, control, optim=FALSE)
```

## Arguments

| | |
|---|---|
| `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. |
| `type` | default "lag", may be set to "mixed"; when "mixed", the lagged intercept is dropped for spatial weights style "W", that is row-standardised weights, but otherwise included |
| `method` | "eigen" (default) - the Jacobian is computed as the product of (1 - rho*eigenvalue) using `eigenw`, and "SparseM" for strictly symmetric weights lists of styles "B", "C" and "U", or made symmetric by similarity (Ord, 1975, Appendix C) if possible for styles "W" and "S", using code from the SparseM package to calculate the determinant. |
| `quiet` | default=TRUE; if FALSE, reports function values during optimization. |
| `zero.policy` | if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `lagsarlm()` to terminate with an error |
| `interval` | search interval for autoregressive parameter when not using method="eigen"; default is c(-1,1) |

| | |
|---|---|
| tol.solve | the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to solve() (default=1.0e-10). This may be used if necessary to extract coefficient standard errors (for instance lowering to 1e-12), but 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 |
| tol.opt | the desired accuracy of the optimization - passed to optimize() (default=square root of double precision machine tolerance) |
| control | A list of control parameters passed to optim, se details in optim |
| optim | If TRUE use experimental optim branch and control argument |

## Details

The asymptotic standard error of $\rho$ is only computed when method=eigen, because the full matrix operations involved would be costly for large n typically associated with the choice of method="SparseM". The same applies to the coefficient covariance matrix. Taken as the asymptotic matrix from the literature, it is typically badly scaled, and with the elements involving rho being very small, while other parts of the matrix can be very large (often many orders of magnitude in difference). It often happens that the tol.solve argument needs to be set to a smaller value than the default, or the RHS variables can be centred or reduced in range.

## Value

A list object of class sarlm

| | |
|---|---|
| type | "lag" or "mixed" |
| rho | simultaneous autoregressive lag coefficient |
| coefficients | GLS coefficient estimates |
| rest.se | asymptotic standard errors if ase=TRUE |
| LL | log likelihood value at computed optimum |
| s2 | GLS residual variance |
| SSE | sum of squared GLS errors |
| parameters | number of parameters estimated |
| lm.model | the lm object returned when estimating for $\rho = 0$ |
| method | the method used to calculate the Jacobian |
| call | the call used to create this object |
| residuals | GLS residuals |
| lm.target | the lm object returned for the GLS fit |
| fitted.values | Difference between residuals and response variable |
| se.fit | Not used yet |
| formula | model formula |
| ase | TRUE if method=eigen |
| LLs | if ase=FALSE (for method="SparseM"), the log likelihood values of models estimated dropping each of the independent variables in turn, used in the summary function as a substitute for variable coefficient significance tests |

| | |
|---|---|
| `rho.se` | if ase=TRUE, the asymptotic standard error of $\rho$ |
| `LMtest` | if ase=TRUE, the Lagrange Multiplier test for the absence of spatial autocorrelation in the lag model residuals |
| `zero.policy` | zero.policy for this model |
| `na.action` | (possibly) named vector of excluded or omitted observations if non-default na.action argument used |

The internal sar.lag.mixed.* functions return the value of the log likelihood function at $\rho$.

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩, with thanks to Andrew Bernat for contributions to the asymptotic standard error code.

## References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion; Ord, J. K. 1975 Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, 70, 120-126; Anselin, L. 1988 *Spatial econometrics: methods and models.* (Dordrecht: Kluwer); Anselin, L. 1995 SpaceStat, a software program for the analysis of spatial data, version 1.80. Regional Research Institute, West Virginia University, Morgantown, WV (www.spacestat.com); Anselin L, Bera AK (1998) Spatial dependence in linear regression models with an introduction to spatial econometrics. In: Ullah A, Giles DEA (eds) Handbook of applied economic statistics. Marcel Dekker, New York, pp. 237-289.

## See Also

`lm`, `errorsarlm`, `eigenw`, `asMatrixCsrListw`, `predict.sarlm`, `residuals.sarlm`

## Examples

```
data(oldcol)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb), method="eigen", quiet=FALSE)
COL.lag.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", quiet=FALSE)
COL.lag.SM <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb), method="SparseM", quiet=FALSE)
summary(COL.lag.eig, correlation=TRUE)
summary(COL.lag.SM, correlation=TRUE)
COL.lag.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="B"))
summary(COL.lag.B, correlation=TRUE)
COL.mixed.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="B"), type="mixed", tol.solve=1e-9)
summary(COL.mixed.B, correlation=TRUE)
COL.mixed.W <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), type="mixed")
summary(COL.mixed.W, correlation=TRUE)
NA.COL.OLD <- COL.OLD
NA.COL.OLD$CRIME[20:25] <- NA
COL.lag.NA <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 nb2listw(COL.nb), na.action=na.exclude, tol.opt=.Machine$double.eps^0.4)
COL.lag.NA$na.action
COL.lag.NA
```

```
resid(COL.lag.NA)
```

---

`listw2sn` *Spatial neighbour sparse representation*

---

## Description

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

## Usage

```
listw2sn(listw)
sn2listw(sn)
```

## Arguments

| | |
|---|---|
| `listw` | a `listw` object from for example `nb2listw` |
| `sn` | a `spatial.neighbour` object |

## Value

A data frame with three columns, and with class `spatial.neighbour`:

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

`logSpwdet` returns log det(I - rho * W).

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## See Also

`nb2listw`, `errorsarlm`, `lagsarlm`

## Examples

```
data(columbus)
col.listw <- nb2listw(col.gal.nb)
col.listw$neighbours[[1]]
col.listw$weights[[1]]
col.sn <- listw2sn(col.listw)
str(col.sn)
W <- asMatrixCsrListw(similar.listw(col.listw))
str(W)
rho <- seq(-0.8, 0.9, 0.1)
for (i in rho) print(paste("rho:", i, "log(det(I - rho*W))",
  log(det(asMatrixCsrIrW(W, i)))), quote=TRUE)
```

`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).

## Usage

```
lm.LMtests(model, listw, zero.policy=FALSE, test="LMerr", spChk=NULL)
print.LMtestlist(x, ...)
```

## Arguments

| | |
|---|---|
| `model` | an object of class `lm` returned by `lm`; weights and offsets should not be used |
| `listw` | a `listw` object created for example by `nb2listw`, expected to be row-standardised (W-style) |
| `zero.policy` | if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA |
| `test` | a character vector of tests requested chosen from LMerr, LMlag, RLMerr, RLMlag, SARMA; test="all" computes all the tests. |
| `spChk` | should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()` |
| `x` | object to be printed |
| `...` | 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 `htest` objects, each with:

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

### 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)
lm.LMtests(oldcrime.lm, nb2listw(COL.nb), test=c("LMerr", "LMlag", "RLMerr",
  "RLMlag", "SARMA"))
```

---

## `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=FALSE,
            alternative = "greater", spChk=NULL, resfun=weighted.residuals)
listw2U(listw)
```

### 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 | 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 |

### Value

A list with class htest containing the following components:

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

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

### See Also

lm.LMtests, lm

### Examples

```
data(oldcol)
oldcrime1.lm <- lm(CRIME ~ 1, data = COL.OLD)
oldcrime.lm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD)
lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="W"))
lm.LMtests(oldcrime.lm, nb2listw(COL.nb, style="W"))
lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="S"))
lm.morantest(oldcrime1.lm, nb2listw(COL.nb, style="W"))
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
oldcrime.wlm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD, weights = I(1/AREA))
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.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=FALSE, alternative="greater",
  spChk=NULL, resfun=weighted.residuals, tol=.Machine$double.eps^0.5, maxiter=10
print.moransad(x, ...)
summary.moransad(object, ...)
print.summary.moransad(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 | 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 |
| x | object to be printed |
| object | object to be summarised |
| ... | arguments to be passed through |

### Details

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

## Value

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

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

Tiefelsdorf, M. 2002 The Saddlepoint approximation of Moran's I and local Moran's Ii reference distributions and their numerical evaluation. Geographical Analysis, forthcoming; see also Tiefelsdorf's SPSS code: http://geog-www.sbs.ohio-state.edu/faculty/tiefelsdorf/GeoStat.htm.

## See Also

`lm.morantest`

## Examples

```
data(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire.df)
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.df, 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 posibility of a local cluster of high values of the variable being analysed, very low relative values a similar cluster of low values. For inference, a Bonferroni-type test is suggested in the references, where tables of critical values may be found (see also details below).

## Usage

```
localG(x, listw, zero.policy=FALSE, spChk=NULL)
```

## 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` | 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()` |

## 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 in the spweights package 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.

**Examples**

```
data(getisord)
xycoords <- cbind(xyz$x, xyz$y)
nb30 <- dnearneigh(xycoords, 0, 30)
G30 <- localG(spNamedVec("val", xyz), nb2listw(nb30, style="B"))
G30[length(xyz$val)-136]
nb60 <- dnearneigh(xycoords, 0, 60)
G60 <- localG(spNamedVec("val", xyz), nb2listw(nb60, style="B"))
G60[length(xyz$val)-136]
nb90 <- dnearneigh(xycoords, 0, 90)
G90 <- localG(spNamedVec("val", xyz), nb2listw(nb90, style="B"))
G90[length(xyz$val)-136]
nb120 <- dnearneigh(xycoords, 0, 120)
G120 <- localG(spNamedVec("val", xyz), nb2listw(nb120, style="B"))
G120[length(xyz$val)-136]
nb150 <- dnearneigh(xycoords, 0, 150)
G150 <- localG(spNamedVec("val", xyz), 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$y, round(G30, digits=1), cex=0.7)
polygon(c(195,225,225,195), c(195,195,225,225), lwd=2)
title(main=expression(paste("Values of the ", G[i], " statistic")))
G30s <- localG(spNamedVec("val", xyz), nb2listw(include.self(nb30),
 style="B"))
cat("value according to Getis and Ord's eq. 14.2, p. 263 (1996)\n")
G30s[length(xyz$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:

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

, and its expectation and variance are given in Anselin (1995).

### Usage

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

### 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 | 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 p.adjustSP. Note that the number of multiple tests for each region is only taken as the number of neighbours + 1 for each region, rather than the total number of regions. |
| spChk | should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption() |

### 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.

### See Also

localG

### Examples

```
data(afcon)
oid <- order(afcon$id)
resI <- localmoran(spNamedVec("totcon", afcon), nb2listw(paper.nb))
printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]), check.names=FALSE)
hist(resI[,5])
resI <- localmoran(spNamedVec("totcon", afcon), nb2listw(paper.nb), p.adjust.method="bonf
printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]), check.names=FALSE)
hist(resI[,5])
totcon <- spNamedVec("totcon", afcon)
is.na(totcon) <- sample(1:length(totcon), 5)
totcon
resI.na <- localmoran(totcon, nb2listw(paper.nb), na.action=na.exclude,
 zero.policy=TRUE)
if (class(attr(resI.na, "na.action")) == "exclude") {
 print(data.frame(resI.na[oid,], row.names=afcon$name[oid]), digits=2)
} else print(resI.na, digits=2)
resG <- localG(spNamedVec("totcon", afcon), nb2listw(include.self(paper.nb)))
print(data.frame(resG[oid], row.names=afcon$name[oid]), digits=2)
```

---

## 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=FALSE,
  alternative="greater", spChk=NULL, resfun=weighted.residuals, save.Vi=FALSE,
  tol = .Machine$double.eps^0.5, maxiter = 1000, tol.bounds=0.0001)
as.data.frame.localmoransad(x, row.names=NULL, optional=FALSE)
print.localmoransad(x, ...)
summary.localmoransad(object, ...)
print.summary.localmoransad(x, ...)
listw2star(listw, ireg, style, n, D, a, zero.policy=FALSE)
```

### Arguments

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

| | |
|---|---|
| select | an integer vector of the id. numbers of zones to be tested |
| nb | a list of neighbours of class nb |
| glist | a list of general weights corresponding to neighbours |
| style | can take values W, B, C, and S |
| zero.policy | if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA |
| alternative | a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided. |
| spChk | should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption() |
| resfun | default: weighted.residuals; the function to be used to extract residuals from the lm object, may be residuals, weighted.residuals, rstandard, or rstudent |
| save.Vi | if TRUE, return the star-shaped weights lists for each zone tested |
| tol | the desired accuracy (convergence tolerance) for uniroot |
| maxiter | the maximum number of iterations for uniroot |
| tol.bounds | offset from bounds for uniroot |
| x | object to be printed |
| row.names | ignored argument to as.data.frame.localmoransad; row names assigned from localmoransad object |
| optional | ignored argument to as.data.frame.localmoransad; row names assigned from localmoransad object |
| object | object to be summarised |
| ... | arguments to be passed through |
| listw | a listw object created for example by nb2listw |
| ireg | a zone number |
| n | internal value depending on listw and style |
| D | internal value depending on listw and style |
| a | internal value depending on listw and style |

**Details**

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

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

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

## Value

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

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

Tiefelsdorf, M. 2002 The Saddlepoint approximation of Moran's I and local Moran's Ii reference distributions and their numerical evaluation. Geographical Analysis, 34, pp. 187–206; see also Tiefelsdorf's SPSS code: http://geog-www.sbs.ohio-state.edu/faculty/tiefelsdorf/GeoStat.htm.

## See Also

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

## Examples

```
data(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire.df)
e.locmor <- summary(localmoran.sad(e.lm, eire.nb, select=1:nrow(eire.df)))
e.locmor
mean(e.locmor[,1])
lm.morantest(e.lm, nb2listw(eire.nb))
hist(e.locmor[,"Pr. (Sad)"])
e.wlm <- lm(OWNCONS ~ ROADACC, data=eire.df, weights=RETSALE)
e.locmorw1 <- summary(localmoran.sad(e.wlm, eire.nb, select=1:nrow(eire.df), resfun=weigh
e.locmorw1
e.locmorw2 <- summary(localmoran.sad(e.wlm, eire.nb, select=1:nrow(eire.df), resfun=rstuc
e.locmorw2
e.errorsar <- errorsarlm(OWNCONS ~ ROADACC, data=eire.df,
 listw=nb2listw(eire.nb))
e.errorsar
e.clocmor <- summary(localmoran.sad(e.errorsar, eire.nb, select=1:nrow(eire.df)))
e.clocmor
hist(e.clocmor[,"Pr. (Sad)"])
```

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

## Description

The function converts a square spatial weights 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.

## Usage

```
mat2listw(x, row.names = NULL)
```

## Arguments

| | |
|---|---|
| `x` | A square non-negative matrix with no NAs representing spatial weights |
| `row.names` | row names to use for region IDs |

## Value

A `listw` object with the following members:

| | |
|---|---|
| `style` | "M", meaning matrix style, underlying style unknown |
| `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

```
data(columbus)
col005 <- dnearneigh(coords, 0, 0.5, attr(col.gal.nb, "region.id"))
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)
```

`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=FALSE, NAOK=FALSE)
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector the same length as the neighbours list in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `n` | number of zones |
| `S0` | global sum of weights |
| `zero.policy` | 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 <- spNamedVec("CRIME", COL.OLD)
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. The examples show how `boot(sim="permutation")` can replicate this function (thanks to Virgilio Gómez Rubio and the DCluster package).

## Usage

```
moran.mc(x, listw, nsim, zero.policy=FALSE, alternative="greater", na.action=na.
```

## Arguments

| | |
|---|---|
| `x` | a numeric vector the same length as the neighbours list in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `nsim` | number of permutations |
| `zero.policy` | 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". |
| `na.action` | a function (default `na.fail`), can also be `na.omit` or `na.exclude` - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted. `na.pass` is not permitted because it is meaningless in a permutation test. |
| `spChk` | should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()` |

## Value

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

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

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

### See Also

moran, moran.test

### Examples

```
data(oldcol)
colw <- nb2listw(COL.nb, style="W")
nsim <- 99
set.seed(1234)
sim1 <- moran.mc(spNamedVec("CRIME", COL.OLD), listw=colw, nsim=nsim)
sim1
mean(sim1$res[1:nsim])
var(sim1$res[1:nsim])
summary(sim1$res[1:nsim])
MoranI.boot <- function(var, i, ...) {
var <- var[i]
return(moran(x=var, ...)$I)
}
set.seed(1234)
library(boot)
boot1 <- boot(spNamedVec("CRIME", COL.OLD), statistic=MoranI.boot, R=nsim, sim="permutati
boot1
plot(boot1)
mean(boot1$t)
var(boot1$t)
summary(boot1$t)
colold.lags <- nblag(COL.nb, 3)
set.seed(1234)
sim2 <- moran.mc(spNamedVec("CRIME", COL.OLD), nb2listw(colold.lags[[2]],
 style="W"), nsim=nsim)
summary(sim2$res[1:nsim])
sim3 <- moran.mc(spNamedVec("CRIME", COL.OLD), 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=FALSE, spChk=NULL, labels=NULL,
 xlab=NULL, ylab=NULL, quiet=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 | 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 | if TRUE, output of summary of influence object suppressed |
| ... | further graphical parameters as in par(..) |

### Value

The function returns 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, summary.infl

### Examples

```
data(afcon)
moran.plot(spNamedVec("totcon", afcon), nb2listw(paper.nb),
 labels=as.character(afcon$name), pch=19)
moran.plot(scale(spNamedVec("totcon", afcon)), nb2listw(paper.nb),
 labels=as.character(afcon$name), xlim=c(-2, 4), ylim=c(-2,4), pch=19)
```

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

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `x` | a numeric vector the same length as the neighbours list in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `randomisation` | variance of I calculated under the assumption of randomisation, if FALSE normality |
| `zero.policy` | if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA |
| `alternative` | a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided. |
| `rank` | logical value - default FALSE for continuous variables, if TRUE, uses the adaptation of Moran's I for ranks suggested by Cliff and Ord (1981, p. 46) |
| `na.action` | a function (default `na.fail`), can also be `na.omit` or `na.exclude` - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted. If `na.pass` is used, zero is substituted for NA values in calculating the spatial lag |
| `spChk` | should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()` |

## Value

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

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

### Note

Var(I) is taken from Goodchild's CATMOG 47, see also Upton & Fingleton (1985) p. 171; it agrees with SpaceStat, see Tutorial workbook Chapter 22; VI is as given by Cliff and Ord minus the square of EI. 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.

### See Also

`moran`, `moran.mc`, `listw2U`

### Examples

```
data(oldcol)
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="W"))
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="B"))
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="C"))
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="S"))
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
colold.lags <- nblag(COL.nb, 3)
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(colold.lags[[2]],
 style="W"))
moran.test(spNamedVec("CRIME", COL.OLD), 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(spNamedVec("CRIME", COL.OLD), nb2listw(COL.k4.nb, style="W"))
moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.k4.nb, style="W"),
 randomisation=FALSE)
cat("Note: non-symmetric weights matrix, use listw2U()")
moran.test(spNamedVec("CRIME", COL.OLD), listw2U(nb2listw(COL.k4.nb,
 style="W")))
moran.test(spNamedVec("CRIME", COL.OLD), 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 <- spNamedVec("CRIME", COL.OLD)
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)
```

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

## Description

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

## Usage

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

## Arguments

| | |
|---|---|
| `nb` | an object of class `nb` with a list of integer vectors containing neighbour region number ids |
| `ID` | identification tags of the locations for the location-less observations; `sort(unique(as.charact` must be identical to `sort(as.character(attr(nb, "region.id")))`; same length as `row.names` if provided. |
| `row.names` | character vector of observation ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))`; same length as `ID` if provided. |

## Details

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

## Value

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## See Also

`knn2nb`, `dnearneigh`, `cell2nb`, `tri2nb`, `poly2nb`

### Examples

```
data(boston)
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]
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$TOWN))
block.nb
print(is.symmetric.nb(block.nb))
plot(block.nb, boston.utm)
points(boston.utm, pch=18, col="lightgreen")
moran.test(boston.c$CMEDV, nb2listw(boston.soi))
moran.test(boston.c$CMEDV, nb2listw(block.nb))
```

---

## nb2lines *Use arc-type shapefiles for import and export of weights*

---

### Description

Use arc-type shapefiles 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)
listw2lines(listw, coords)
df2sn(df, i="i", i_ID="i_ID", j="j", wt="wt")
```

### Arguments

| | |
|---|---|
| nb | a neighbour object of class nb |
| wts | list of general weights corresponding to neighbours |
| coords | matrix of region point coordinates |
| listw | a listw object of spatial weights |
| df | a data frame read from a shapefile, derived from the output of nb2lines |
| i | character name of column in df with from entity index |
| i_ID | character name of column in df with from entity region ID |
| j | character name of column in df with to entity index |
| wt | character name of column in df with weights |

### Details

The maptools package function write.linelistShape is used to transport out the list of lines made by nb2lines or listw2lines, which is a simple wrapper function. The neighbour and weights objects may be retrieved by converting the specified columns of the attribute data of the Map object into a spatial.neighbour object, which is then converted into a weights list object.

### Value

`nb2lines` and `listw2lines` return a list of two objects, `ll` is a list of lines, and `df` is a data frame 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`, `write.linelistShape`

### Examples

```
data(columbus)
res <- listw2lines(nb2listw(col.gal.nb), coords)
str(res$df)
fn <- paste(tempdir(), "nbshape", sep="/")
write.linelistShape(res$ll, res$df, file=fn)
inMap <- read.shape(fn)
str(inMap$att.data)
diffnb(sn2listw(df2sn(inMap$att.data))$neighbours, col.gal.nb)
identical(coords, unique(t(sapply(Map2lines(inMap), function(x) x[1,]))))
```

---

## `nb2listw` *Spatial weights for neighbours lists*

---

### Description

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

### Usage

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

### Arguments

| | |
|---|---|
| `neighbours` | an object of class `nb` |
| `glist` | list of general weights corresponding to neighbours |
| `style` | `style` can take values W, B, C, U, and S |
| `zero.policy` | If FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors |

## Details

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

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

## Value

A `listw` object with the following members:

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

## Author(s)

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

`summary.nb`, `read.gal`

## Examples

```
data(columbus)
cards <- card(col.gal.nb)
col.w <- nb2listw(col.gal.nb)
plot(cards, unlist(lapply(col.w$weights, sum)),xlim=c(0,10),
ylim=c(0,10), xlab="number of links", ylab="row sums of weights")
col.b <- nb2listw(col.gal.nb, style="B")
points(cards, unlist(lapply(col.b$weights, sum)), col="red")
col.c <- nb2listw(col.gal.nb, style="C")
points(cards, unlist(lapply(col.c$weights, sum)), col="green")
col.u <- nb2listw(col.gal.nb, style="U")
points(cards, unlist(lapply(col.u$weights, sum)), col="orange")
col.s <- nb2listw(col.gal.nb, style="S")
points(cards, unlist(lapply(col.s$weights, sum)), col="blue")
legend(x=c(0, 1), y=c(7, 9), legend=c("W", "B", "C", "U", "S"),
col=c("black", "red", "green", "orange", "blue"), pch=rep(1,5))
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))
```

---

`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=FALSE)
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` | If FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors |
| `listw` | a `listw` object from for example `nb2listw` |

## Details

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

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

## Value

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

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

### See Also

nb2listw

### Examples

```
data(columbus)
col005 <- dnearneigh(coords, 0, 0.5, attr(col.gal.nb, "region.id"))
summary(col005)
col005.w.mat <- nb2mat(col005, zero.policy=TRUE)
table(round(apply(col005.w.mat, 1, sum)))
```

---

## 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 lonlat = TRUE, Great Circle distances are used.

### Usage

```
nbdists(nb, coords, lonlat = FALSE)
```

### Arguments

| | |
|---|---|
| nb | an object of class nb |
| coords | matrix of point coordinates |
| lonlat | TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers |

### Value

A list with class nbdist

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

summary.nb, nb2listw

### Examples

```
data(columbus)
dlist <- nbdists(col.gal.nb, coords)
dlist <- lapply(dlist, function(x) 1/x)
stem(unlist(dlist))
```

`nblag` *Higher order neighbours lists*

## Description

The function creates higher order neighbour lists, where higher order neighbours are only `lags` links from each other on the graph described by the input neighbours list. It will refuse to lag neighbours lists with the attribute self.included set to TRUE.

## Usage

```
nblag(neighbours, maxlag)
```

## Arguments

| | |
|---|---|
| `neighbours` | input neighbours list of class `nb` |
| `maxlag` | the maximum lag to be constructed |

## Value

returns a list of lagged neighbours lists each with class `nb`

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## See Also

`summary.nb`

## Examples

```
data(columbus)
summary(col.gal.nb, coords)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(col.gal.nb, coords, add=TRUE)
title(main="GAL order 1 (black) and 2 (red) links")
col.lags <- nblag(col.gal.nb, 2)
summary(col.lags[[2]], coords)
plot(col.lags[[2]], coords, add=TRUE, col="red", lty=2)
```

## 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 ⟨kohnicho@comp.nus.edu.sg⟩

### See Also

intersect.nb, union.nb, setdiff.nb

### Examples

```
data(columbus)
col.tri.nb <- tri2nb(coords)
oldpar <- par(mfrow=c(1,2))
col.soi.nb <- graph2nb(soi.graph(col.tri.nb, coords))
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(col.soi.nb, coords, add=TRUE)
title(main="Sphere of Influence Graph")
plot(polys, border="grey", forcefill=FALSE)
plot(complement.nb(col.soi.nb), coords, add=TRUE)
title(main="Complement of Sphere of Influence Graph")
par(mfrow=c(2,2))
col2 <- droplinks(col.gal.nb, 21)
```

```
plot(intersect.nb(col.gal.nb, col2), coords)
title(main="Intersect")
plot(union.nb(col.gal.nb, col2), coords)
title(main="Union")
plot(setdiff.nb(col.gal.nb, col2), coords)
title(main="Set diff")
par(oldpar)
```

---

`nc.sids` *North Carolina SIDS data*

---

## Description

The `nc.sids` data frame has 100 rows and 9 columns. It contains data given in Cressie (1991, pp. 386-9), Cressie and Read (1985) and Cressie and Chan (1989) on sudden infant deaths in North Carolina for 1974-78 and 1979-84. The data set also contains the neighbour list given by Cressie and Chan (1989) omitting self-neighbours (ncCC89.nb), and the neighbour list given by Cressie and Read (1985) for contiguities (ncCR85.nb). The data are ordered by county ID number, not alphabetically as in the source tables `sidspolys` is a "polylist" object of polygon boundaries, and `sidscents` is a matrix of their centroids.

## Usage

```
data(nc.sids)
```

## Format

This data frame contains the following columns:

**CNTY.ID** county ID

**BIR74** births, 1974-78

**SID74** SID deaths, 1974-78

**NWBIR74** non-white births, 1974-78

**BIR79** births, 1979-84

**SID79** SID deaths, 1979-84

**NWBIR79** non-white births, 1979-84

**east** eastings, county seat, miles, local projection

**north** northings, county seat, miles, local projection

**x** easting, county seats approximately reprojected to UTM zone 18

**y** northings, county seats approximately reprojected to UTM zone 18

**lon** easting, county seats in long-lat

**lat** northings, county seats in long-lat

**L.id** Cressie and Read (1985) L index

**M.id** Cressie and Read (1985) M index

### Source

Cressie, N (1991), *Statistics for spatial data*. New York: Wiley, pp. 386–389; Cressie, N, Chan NH (1989) Spatial modelling of regional variables. *Journal of the American Statistical Association*, 84, 393–401; Cressie, N, Read, TRC (1985) Do sudden infant deaths come in clusters? *Statistics and Decisions* Supplement Issue 2, 333–349; http://sal.agecon.uiuc.edu/datasets/sids.zip.

### Examples

```
library(maptools)
data(nc.sids)
plot(sidspolys, border="grey", forcefill=FALSE)
plot(ncCR85.nb, sidscents, add=TRUE, col="blue")
plot(sidspolys, border="grey", forcefill=FALSE)
plot(ncCC89.nb, cbind(nc.sids$lon, nc.sids$lat),
 add=TRUE, col="blue")
```

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

Danlin Yu and Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

p.adjust, localG, localmoran

### Examples

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

---

## plot.nb *Plot a neighbours list*

---

### Description

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

### Usage

```
plot.nb(x, coords, col="black", points=TRUE, add=FALSE, arrows=FALSE, length=0.1
plot.listw(x, coords, col="black", points=TRUE, add=FALSE, arrows=FALSE, length=
```

### Arguments

| | |
|---|---|
| `x` | an object of class `nb` or (for `plot.listw`) class `listw` |
| `coords` | matrix of region point coordinates |
| `col` | plotting colour |
| `points` | (logical) add points to plot |
| `add` | (logical) add to existing plot |
| `arrows` | (logical) draw arrowheads for asymmetric neighbours |
| `length` | length in plot inches of arrow heads drawn for asymmetric neighbours lists |
| `...` | further graphical parameters as in `par(..)` |

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

`summary.nb`

### Examples

```
data(columbus)
plot(col.gal.nb, coords)
title(main="GAL order 1 links with first nearest neighbours in red")
col.knn <- knearneigh(coords, k=1)
plot(knn2nb(col.knn), coords, add=TRUE, col="red", length=0.08)
```

`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)
```

## Arguments

| | |
|---|---|
| `pl` | list of polygons of class `polylist` |
| `row.names` | character vector of region ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))`; if `polys` has a `region.id` attribute, it is copied to the neighbours list. |
| `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 |

## Value

A neighbours list with class `nb`

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## See Also

`summary.nb`, `plot.polylist`

## Examples

```
data(columbus)
xx <- poly2nb(polys)
dxx <- diffnb(xx, col.gal.nb)
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(col.gal.nb, coords, add=TRUE)
plot(dxx, coords, add=TRUE, col="red")
title(main="Differences (red) in Columbus GAL weights (black)\nand polygon generated quee
xxx <- poly2nb(polys, queen=FALSE)
dxxx <- diffnb(xxx, col.gal.nb)
plot(polys, border = "grey", forcefill=FALSE)
plot(col.gal.nb, coords, add = TRUE)
plot(dxxx, coords, add = TRUE, col = "red")
```

```
title(main="Differences (red) in Columbus GAL weights (black)\nand polygon generated rook
cards <- card(xx)
maxconts <- which(cards == max(cards))
if(length(maxconts) > 1) maxconts <- maxconts[1]
fg <- rep("grey", length(polys))
fg[maxconts] <- "red"
fg[xx[[maxconts]]] <- "green"
plot(polys, col=fg, forcefill=FALSE)
title(main="Region with largest number of contiguities")
```

---

`predict.sarlm` *Prediction for spatial simultaneous autoregressive linear model objects*

---

## Description

`predict.sarlm()` calculates predictions as far as is at present possible for for spatial simultaneous autoregressive linear model objects, using Haining's terminology for decomposition into trend, signal, and noise — see reference.

## Usage

```
predict.sarlm(object, newdata = NULL, listw = NULL, zero.policy = FALSE, ...)
print.sarlm.pred(x, ...)
```

## Arguments

| | |
|---|---|
| `object` | `sarlm` object returned by `lagsarlm` or `errorsarlm` |
| `newdata` | Data frame in which to predict — if NULL, predictions are for the data on which the model was fitted |
| `listw` | a `listw` object created for example by `nb2listw` |
| `zero.policy` | if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `lagsarlm()` to terminate with an error |
| `x` | the object to be printed |
| `...` | further arguments passed through |

## Details

In the following, the trend is the non-spatial smooth, the signal is the spatial smooth, and the noise is the residual. The fit returned is the sum of the trend and the signal.

The function approaches prediction first by dividing invocations between those with or without newdata. When no newdata is present, the response variable may be reconstructed as the sum of the trend, the signal, and the noise (residuals). Since the values of the response variable are known, their spatial lags are used to calculate signal components. For the error model, trend = $X\beta$, and signal = $\lambda Wy - \lambda WX\beta$. For the lag and mixed models, trend = $X\beta$, and signal = $\rho Wy$.

When however newdata is used for prediction, no observations of the response variable being predicted are available. Consequently, while the trend components are the same, the signal cannot take full account of the spatial smooth. In the error model, the signal is set to zero, since the spatial smooth is expressed in terms of the error: $(I - \lambda W)^{-1}\varepsilon$.

In the lag model, the signal can be expressed in the following way:

$$(I - \rho W)y = X\beta + \varepsilon$$

,

$$y = (I - \rho W)^{-1}X\beta + (I - \rho W)^{-1}\varepsilon$$

giving a feasible signal component of:

$$\rho W y = \rho W (I - \rho W)^{-1} X\beta$$

setting the error term to zero. This also means that predictions of the signal component for lag and mixed models require the inversion of an n-by-n matrix.

Because the outcomes of the spatial smooth on the error term are unobservable, this means that the signal values for newdata are incomplete. In the mixed model, the spatially lagged RHS variables influence both the trend and the signal, so that the root mean square prediction error in the examples below for this case with newdata is smallest, although the model was not the best fit

## Value

`predict.sarlm()` returns a vector of predictions with two attribute vectors of trend and signal values with class `sarlm.pred`. `print.sarlm.pred` is a print function for this class, printing and returning a data frame with columns: "fit", "trend" and "signal".

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

Haining, R. 1990 *Spatial data analysis in the social and environmental sciences*, Cambridge: Cambridge University Press, p. 258.

## See Also

`errorsarlm`, `lagsarlm`

## Examples

```
data(oldcol)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
COL.mix.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb),
  type="mixed")
COL.err.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
print(p1 <- predict(COL.mix.eig))
print(p2 <- predict(COL.mix.eig, newdata=COL.OLD, listw=nb2listw(COL.nb)))
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))
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=nb2listw(COL.nb))))^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=nb2listw(COL.nb))))^2)/length(COL.nb))
```

---

`probmap` *Probability mapping for rates*

---

## Description

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

## Usage

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

## 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 |

## Details

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

## Value

| | |
|---|---|
| `raw` | raw (crude) rates |
| `expCount` | expected counts of cases assuming global rate |
| `relRisk` | relative risks: ratio of observed and expected counts of cases multiplied by 100 |
| `pmap` | Poisson probability map values: probablility of getting a more “extreme” count than actually observed - here two-tailed, with extreme tails indicating “unusual” values |

## 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

```
data(auckland)
res <- probmap(auckland$Deaths.1977.85, 9*auckland$Under.5.1981)
brks <- c(-Inf,2,2.5,3,3.5,Inf)
cols <- grey(6:2/7)
library(maptools)
plot(auckpolys, col=cols[findInterval(res$raw*1000, brks)], forcefill=FALSE)
legend(c(70,90), c(70,95), fill=cols, legend=leglabs(brks), bty="n")
title(main="Crude (raw) estimates of infant mortality per 1000 per year")
brks <- c(-Inf,47,83,118,154,190,Inf)
cols <- cm.colors(6)
plot(auckpolys, col=cols[findInterval(res$relRisk, brks)], forcefill=FALSE)
legend(c(70,90), c(70,95), fill=cols, legend=leglabs(brks), bty="n")
title(main="Standardised mortality ratios for Auckland child deaths")
brks <- c(0,0.05,0.1,0.2,0.8,0.9,0.95,1)
cols <- cm.colors(7)
plot(auckpolys, col=cols[findInterval(res$pmap, brks)], forcefill=FALSE)
legend(c(70,90), c(70,95), fill=cols, legend=leglabs(brks), bty="n")
title(main="Poisson probabilities for Auckland child mortality")
```

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Details

Luc Anselin (2003): Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, http://sal.agecon.uiuc.edu/weights/howto.html; 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://agec221.agecon.uiuc.edu/csiss/pdf/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 downloaded from: 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).

### Usage

```
read.gwt2nb(file, region.id=NULL)
write.sn2gwt(sn, file, shpfile=NULL, ind=NULL)
read.dat2listw(file)
write.sn2dat(sn, file)
```

### Arguments

| | |
|---|---|
| file | name of file with weights data |
| region.id | region IDs |
| sn | a spatial.neighbour object |
| shpfile | Shapefile name taken from GWT file for this dataset |
| ind | region id indicator variable name |

### Details

Now attempts to honour the region.id argument given when reading GWT files.

### 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://agec221.agecon.uiuc.edu/csiss/pdf/geoda093.pdf; also http://spatial-econometrics.com/data/contents.html

### See Also

read.gal

### Examples

```
data(baltimore)
STATION <- baltimore$STATION
gwt1 <- read.gwt2nb(system.file("etc/weights/baltk4.GWT", package="spdep")[1],
 STATION)
cat(paste("Neighbours list symmetry;", is.symmetric.nb(gwt1, FALSE, TRUE),
 "\n"))
listw1 <- nb2listw(gwt1, style="B", glist=attr(gwt1, "GeoDa")$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"))
data(oldcol)
diffnb(gwt1, gwt2)
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)
```

---

`residuals.sarlm` *Access functions for spatial simultaneous autoregressive linear model objects*

---

## Description

Access functions for residuals, deviance, coefficients and fitted values from spatial simultaneous autoregressive linear model objects

## Usage

```
residuals.sarlm(object, ...)
deviance.sarlm(object, ...)
coef.sarlm(object, ...)
fitted.sarlm(object, ...)
```

## Arguments

| | |
|---|---|
| `object` | `sarlm` object returned by `lagsarlm` or `errorsarlm` |
| `...` | further arguments passed through |

## Value

Revelant vectors of numerical values.

## Note

`fitted.sarlm()` returns the difference between `residuals()` for the same object and the response variable; `predict.sarlm()` returns a decomposition into trend and signal for the fit.

## Author(s)

Roger Bivand, ⟨Roger.Bivand@nhh.no⟩

## See Also

`errorsarlm`, `lagsarlm`, `predict.sarlm`

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

## Description

Provides support for checking the mutual integrity of spatial neighbour weights and spatial data.

## Usage

```
set.spChkOption(check)
get.spChkOption()
chkIDs(x, listw)
spNamedVec(var, data)
```

## 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)
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))
print(tmp)
set.spChkOption(oldChk)
get.spChkOption()
```

---

## similar.listw *Create symmetric similar weights lists*

---

### Description

From Ord's 1975 paper, it is known that the Jacobian for SAR models may be found by "symmetrizing" by similarity (the eigenvalues of similar matrices are identical, so the Jacobian is too). This applies only to styles "W" and "S" with underlying symmetric binary neighbour relations or symmetric general neighbour relations (so no k-nearest neighbour relations). The function is invoked automatically within the SAR fitting functions, to call `eigen` on a symmetric matrix for the default eigen method, or to make it possible to use the SparseM method on weights that can be "symmetrized" in this way.

### Usage

```
similar.listw(listw)
```

### Arguments

| | |
|---|---|
| `listw` | a `listw` object created for example by `nb2listw` |

### Value

a `listw` object

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Ord, J. K. 1975 Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, 70, 120-126

### See Also

asMatrixCsrListw, lagsarlm, errorsarlm

### Examples

```
data(oldcol)
COL.W <- nb2listw(COL.nb, style="W")
COL.S <- nb2listw(COL.nb, style="S")
log(prod(1 - 0.5 * eigenw(COL.W)))
log(prod(1 - 0.5 * eigenw(similar.listw(COL.W))))
Det <- getMethod("det", "matrix.csr")
log(Det(asMatrixCsrIrW(asMatrixCsrListw(similar.listw(COL.W)), 0.5)))
log(prod(1 - 0.5 * eigenw(COL.S)))
log(prod(1 - 0.5 * eigenw(similar.listw(COL.S))))
log(Det(asMatrixCsrIrW(asMatrixCsrListw(similar.listw(COL.S)), 0.5)))
```

---

## sp.correlogram *Spatial correlogram*

---

### Description

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

### Usage

```
sp.correlogram(neighbours, var, order = 1, method = "corr", style = "W",
  randomisation = TRUE, zero.policy = FALSE, spChk=NULL)
plot.spcor(x, main, ylab, ylim, ...)
print.spcor(x, ...)
```

### Arguments

| | |
|---|---|
| neighbours | an object of class nb |
| var | a numeric vector |
| order | maximum lag order |
| method | "corr" for correlation, "I" for Moran's I |
| style | style can take values W, B, C, and S |
| randomisation | variance of I calculated under the assumption of randomisation, if FALSE normality |
| zero.policy | 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 |
| 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 |

## 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

## Examples

```
data(nc.sids)
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)
names(tr.SIDS74) <- rownames(nc.sids)
print(sp.correlogram(ncCC89.nb, tr.SIDS74, order=8, method="corr",
 zero.policy=TRUE))
print(sp.correlogram(ncCC89.nb, tr.SIDS74, order=8, method="I",
 zero.policy=TRUE))
plot(sp.correlogram(ncCC89.nb, tr.SIDS74, order=8, method="I",
 zero.policy=TRUE))
plot(sp.correlogram(ncCC89.nb, tr.SIDS74, order=8, method="corr",
 zero.policy=TRUE))
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 = FALSE,
 alternative = "greater", spChk=NULL)
plot.mc.sim(x, ...)
```

## Arguments

| | |
|---|---|
| `var` | a numeric vector the same length as the neighbours list in listw |
| `listw` | a `listw` object created for example by `nb2listw` |
| `nsim` | number of permutations |
| `type` | "moran", "geary" or "sokal" criteria for similarity |
| `zero.policy` | 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()` |
| `x` | the object to be plotted |
| `...` | further arguments passed through |

## Value

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

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

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 22-24, Haining, R. 1990 *Spatial data analysis in the social and environmental sciences*, Cambridge: Cambridge University Press, p. 230–1. The function has been checked against general matrix code posted to the r-help list by Ben Bolker on 1 May 2001; another `mantel()` function is in the vegan package.

### See Also

`moran.mc`, `joincount.mc`, `geary.mc`

### Examples

```
data(oldcol)
sim1 <- sp.mantel.mc(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb),
 nsim=99, type="geary", alternative="less")
sim1
plot(sim1)
sp.mantel.mc(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb), nsim=99,
 type="sokal", alternative="less")
sp.mantel.mc(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb), nsim=99,
 type="moran")
```

---

## spdep *Return package version number*

---

### Description

The function retreives package version and build information

### Usage

```
spdep(build = FALSE)
```

### Arguments

| | |
|---|---|
| `build` | if TRUE, also returns build information |

### Value

a character vector with one or two elements

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

`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

## Usage

```
spweights.constants(listw, zero.policy=FALSE)
Szero(listw)
```

## Arguments

| | |
|---|---|
| `listw` | a `listw` object from for example `nb2listw` |
| `zero.policy` | if TRUE ignore zones without neighbours, if FALSE fail when encountered |

## 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")))
```

---

## 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

```
subset.listw(x, subset, zero.policy = FALSE, ...)
```

### Arguments

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

```
data(columbus)
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

```
subset.nb(x, subset, ...)
```

### Arguments

| | |
|---|---|
| x | an object of class nb |
| subset | logical expression |
| ... | generic function pass-through |

### Value

The function returns an object of class nb with a list of integer vectors containing neighbour region number ids (compacted to run from 1:number of regions in subset).

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

nb2listw

### Examples

```
data(columbus)
plot(col.gal.nb, coords)
to.be.dropped <- c(31, 34, 36, 39, 42, 46)
text(coords[to.be.dropped,1], coords[to.be.dropped,2], labels=to.be.dropped,
  pos=2, offset=0.3)
sub.col.gal.nb <- subset(col.gal.nb,
  !(1:length(col.gal.nb) %in% to.be.dropped))
plot(sub.col.gal.nb, coords[-to.be.dropped,], col="red", add=TRUE)
which(!(attr(col.gal.nb, "region.id") %in%
  attr(sub.col.gal.nb, "region.id")))
```

`summary.nb` *Print and summary function for neighbours and weights lists*

## Description

The function prints summary measures for links in a neighbours list. If a matrix of coordinates is given as well, summary descriptive measures for the link lengths are also printed. Print and summary functions are also available for `"listw"` weights list objects.

## Usage

```
summary.nb(object, coords=NULL, lonlat = FALSE, scale = 1, ...)
print.nb(x, ...)
summary.listw(object, coords, lonlat, zero.policy = FALSE, scale = 1, ...)
print.listw(x, zero.policy = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `object` | an object of class `nb` |
| `coords` | matrix of region point coordinates |
| `lonlat` | TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers |
| `...` | additional arguments affecting the output produced |
| `x` | an object of class `nb` |
| `zero.policy` | 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

```
data(columbus)
col.gal.nb
summary(col.gal.nb, coords)
col.listw <- nb2listw(col.gal.nb, style="W")
col.listw
summary(col.listw)
```

summary.sarlm *summary method for class sarlm*

## Description

Methods used for presenting the results of estimating spatial SAR models.

## Usage

```
summary.sarlm(object, correlation = FALSE, ...)
print.sarlm(x, ...)
print.summary.sarlm(x, digits = max(5, .Options$digits - 3),
        signif.stars = FALSE, ...)
```

## Arguments

| | |
|---|---|
| object | sarlm object from lagsarlm or errorsarlm |
| correlation | logical; if 'TRUE', the correlation matrix of the estimated parameters including sigma is returned and printed (default=FALSE) |
| x | sarlm object from lagsarlm or errorsarlm in print.sarlm, summary object from summary.sarlm for print.summary.sarlm |
| digits | the number of significant digits to use when printing |
| signif.stars | logical. If TRUE, "significance stars" are printed for each coefficient. |
| ... | further arguments passed to or from other methods |

## Value

The summary function summary.sarlm returns the sarlm object augmented with a coefficient matrix with probability values for coefficient asymptotic standard errors for type="error" and for type="lag" or "mixed" when object$ase=TRUE, or a coefficient matrix with probability values for likelihood ratio tests between the model as reported and models with independent variables dropped in turn.

## Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

## References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion; Ord, J. K. 1975 Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, 70, 120-126; Anselin, L. 1988 *Spatial econometrics: methods and models.* (Dordrecht: Kluwer); Anselin, L. 1995 SpaceStat, a software program for the analysis of spatial data, version 1.80. Regional Research Institute, West Virginia University, Morgantown, WV (www.spacestat.com); Anselin L, Bera AK (1998) Spatial dependence in linear regression models with an introduction to spatial econometrics. In: Ullah A, Giles DEA (eds) Handbook of applied economic statistics. Marcel Dekker, New York, pp. 237-289.

## See Also

errorsarlm, lagsarlm, summary.lm

### Examples

```
data(oldcol)
COL.mix.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb), type="mixed", method="eigen", quiet=FALSE)
summary(COL.mix.eig, correlation=TRUE)
```

---

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

---

### Description

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

### Usage

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

### Arguments

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

### Value

TRUE if symmetric, FALSE if not

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

`read.gal`

### Examples

```
data(columbus)
print(is.symmetric.nb(col.gal.nb, verbose=TRUE, force=TRUE))
k4 <- knn2nb(knearneigh(coords, k=4), row.names=rownames(columbus))
k4 <- sym.attr.nb(k4)
print(is.symmetric.nb(k4))
k4.sym <- make.sym.nb(k4)
print(is.symmetric.nb(k4.sym))
```

---

## tri2nb *Neighbours list from tri object*

---

### Description

The function uses the `tripack` package to convert a matrix of two-dimensional coordinates into a neighbours list of class `nb` with a list of integer vectors containing neighbour region number ids.

### Usage

```
tri2nb(coords, row.names = NULL)
```

### Arguments

| | |
|---|---|
| `coords` | matrix of point coordinates with two columns |
| `row.names` | character vector of region ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))` |

### Details

If 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 `cell2nb()` cannot be used.

### Value

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

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

`knn2nb`, `dnearneigh`, `cell2nb`

### Examples

```
require(tripack)
data(columbus)
col.tri.nb <- tri2nb(coords, row.names=rownames(columbus))
library(maptools)
plot(polys, border="grey", forcefill=FALSE)
plot(col.tri.nb, coords, add=TRUE)
title(main="Raw triangulation links")
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,])
```

---

## used.cars *US 1960 used car prices*

---

### Description

The `used.cars` data frame has 48 rows and 2 columns. The data set includes a neighbours list for the 48 states excluding DC from poly2nb().

### Usage

```
data(used.cars)
```

### Format

This data frame contains the following columns:

**tax.charges** taxes and delivery charges for 1955-9 new cars

**price.1960** 1960 used car prices by state

### Source

Hanna, F. A. 1966 Effects of regional differences in taxes and transport charges on automobile consumption, in Ostry, S., Rhymes, J. K. (eds) Papers on regional statistical studies, Toronto: Toronto University Press, pp. 199-223.

### References

Hepple, L. W. 1976 A maximum likelihood model for econometric estimation with spatial series, in Masser, I (ed) Theory and practice in regional science, London: Pion, pp. 90-104.

### Examples

```
data(used.cars)
moran.test(spNamedVec("price.1960", used.cars), nb2listw(usa48.nb))
moran.plot(spNamedVec("price.1960", used.cars), nb2listw(usa48.nb),
  labels=rownames(used.cars))
uc.lm <- lm(price.1960 ~ tax.charges, data=used.cars)
summary(uc.lm)
lm.morantest(uc.lm, nb2listw(usa48.nb))
lm.morantest.sad(uc.lm, nb2listw(usa48.nb))
lm.LMtests(uc.lm, nb2listw(usa48.nb))
uc.err <- errorsarlm(price.1960 ~ tax.charges, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, tol.opt=.Machine$double.eps^0.3)
summary(uc.err)
uc.lag <- lagsarlm(price.1960 ~ tax.charges, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, tol.opt=.Machine$double.eps^0.3)
summary(uc.lag)
uc.lag1 <- lagsarlm(price.1960 ~ 1, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, tol.opt=.Machine$double.eps^0.3)
summary(uc.lag1)
uc.err1 <- errorsarlm(price.1960 ~ 1, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, tol.opt=.Machine$double.eps^0.3)
summary(uc.err1)
```

---

## write.nb.gal *Write a neighbours list as a GAL lattice file*

---

### Description

Write a neighbours list as a GAL lattice file, may also use newer GeoDa header format

### Usage

```
write.nb.gal(nb, file, oldstyle=TRUE, shpfile=NULL, ind=NULL)
```

### Arguments

| | |
|---|---|
| nb | an object of class nb with a list of integer vectors containing neighbour region number ids. |
| file | name of file with GAL lattice data |
| oldstyle | if TRUE, first line of file contains only number of spatial units, if FALSE, uses newer GeoDa style |
| shpfile | Shapefile name taken from GAL file for this dataset |
| ind | region id indicator variable name |

### Author(s)

Roger Bivand ⟨Roger.Bivand@nhh.no⟩

### See Also

read.gal

## Examples

```
data(columbus)
ind <- rownames(columbus)
GALfile <- tempfile("GAL")
write.nb.gal(col.gal.nb, GALfile, oldstyle=FALSE, ind="ind")
col.queen <- read.gal(GALfile, region.id=ind)
unlink(GALfile)
summary(diffnb(col.queen, col.gal.nb))
```

# Index