Package ‘spdep’

March 14, 2011

Version 0.5-29

Date 2011-03-13

Title Spatial dependence: weighting schemes, statistics and models

Encoding latin1

Author Roger Bivand <Roger.Bivand@nhh.no>, with contributions by Micah Altman, Luc Anselin, Renato Assunção, Olaf Berke, Andrew Bernat, Guillaume Blanchet, Eric Blankmeyer, Marilia Carvalho,Bjarke Christensen, Yong-wan Chun, Carsten Dormann, Stéphane Dray, Rein Halbersma, Elias Krainski, Pierre Legendre, Nicholas Lewin-Koh, Hongfei Li, Jielai Ma, Giovanni Millo, Werner Mueller, Hisaji Ono, Pedro Peres-Neto, Gianfranco Piras, Markus Reder, Michael Tiefelsdorf, and Danlin Yu.

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

Depends R (>= 2.10.0), methods, sp (>= 0.9), boot, Matrix (>= 0.999375-9), MASS, nlme, maptools (>= 0.5-4), deldir, coda

Suggests snow, rlecuyer, spam(>= 0.13-1), RANN

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, including regional aggregation by minimum spanning tree; a collection of tests for spatial autocorrelation, including global Moran's I,APLE, 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 and exact tests for global and local Moran's I; and functions for estimating spatial simultaneous autoregressive (SAR) lag and error models,impact measures for lag models, weighted and unweighted SAR and CAR spatial regression models, semi-parametric and Moran eigenvector spatial filtering, GM SAR error models, and generalized spatial two stage least squares models.

License GPL (>= 2)

Repository CRAN

Date/Publication 2011-03-14 10:01:32

R topics documented:

`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(afcon$totcon, nb2listw(africa.rook.nb))
moran.test(afcon$totcon, nb2listw(paper.nb))
geary.test(afcon$totcon, nb2listw(paper.nb))
```

---

aggregate.nb *Aggregate a spatial neighbours object*

Description

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

Usage

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

Arguments

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

Value

an nb neighbour object

Note

Method suggested by Roberto Patuelli

Author(s)

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

Examples

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

`airdist` *Measure distance from plot*

Description

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

Usage

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

Arguments

`ann`	annotate the plot with line measured and distance

Value

a list with members:

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

Author(s)

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

See Also

`locator`

`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

```
example(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)
```

---

aple — *Approximate profile-likelihood estimator (APLE)*

Description

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

Usage

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

Arguments

`x` a zero-mean detrended continuous variable

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

`override\_similarity\_check`
default FALSE, if TRUE - typically for row-standardised weights with asymmetric underlying general weights - similarity is not checked

`useTrace` default TRUE, use trace of sparse matrix `W %*% W` (Li et al. (2010)), if FALSE, use crossproduct of eigenvalues of `W` as in Li et al. (2007)

Details

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

Value

A scalar APLE value.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Li, H, Calder, C. A. and Cressie N. A. C. (2007) Beyond Moran's I: testing for spatial dependence based on the spatial autoregressive model. Geographical Analysis 39, 357-375; Li, H, Calder, C. A. and Cressie N. A. C. (2010) One-step estimation of spatial dependence parameters: Properties and extensions of the APLE statistic, Technical Report No. 846, Department of Statistics, The Ohio State University, Columbus, OH.

See Also

nb2listw, aple.mc, aple.plot

Examples

```
example(wheat)
nbr1 <- poly2nb(wheat, queen=FALSE)
nbrl <- nblag(nbr1, 2)
nbr12 <- nblag_cumul(nbrl)
cms0 <- with(as(wheat, "data.frame"), tapply(yield, c, median))
cms1 <- c(model.matrix(~ factor(c) -1, data=wheat) %*% cms0)
wheat$yield_detrend <- wheat$yield - cms1
isTRUE(all.equal(c(with(as(wheat, "data.frame"),
 tapply(yield_detrend, c, median))), rep(0.0, 25),
 check.attributes=FALSE))
moran.test(wheat$yield_detrend, nb2listw(nbr12, style="W"))
aple(scale(wheat$yield_detrend, scale=FALSE), nb2listw(nbr12, style="W"))
errorsarlm(yield_detrend ~ 1, wheat, nb2listw(nbr12, style="W"))
```

---

aple.mc *Approximate profile-likelihood estimator (APLE) permutation test*

Description

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

Usage

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

Arguments

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

Value

A `boot` object as returned by the `boot` function.

Author(s)

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

References

Li, H, Calder, C. A. and Cressie N. A. C. (2007) Beyond Moran's I: testing for spatial dependence based on the spatial autoregressive model. Geographical Analysis 39, 357-375; Li, H, Calder, C. A. and Cressie N. A. C. (2010) One-step estimation of spatial dependence parameters: Properties and extensions of the APLE statistic, Technical Report No. 846, Department of Statistics, The Ohio State University, Columbus, OH.

See Also

`aple`, `boot`

Examples

```
## Not run:
example(aple)
boot_out <- aple.mc(scale(wheat$yield_detrend, scale=FALSE),
 nb2listw(nbr12, style="W"), nsim=500)
plot(boot_out)
boot_out

## End(Not run)
```

aple.plot — *Approximate profile-likelihood estimator (APLE) scatterplot*

Description

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

Usage

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

Arguments

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

Details

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

Value

`aple.plot` returns list with components:

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

`localAple` returns a vector of local APLE values.

Author(s)

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

References

Li, H, Calder, C. A. and Cressie N. A. C. (2007) Beyond Moran's I: testing for spatial dependence based on the spatial autoregressive model. Geographical Analysis 39, pp. 357-375; Li, H, Calder, C. A. and Cressie N. A. C. (2010) One-step estimation of spatial dependence parameters: Properties and extensions of the APLE statistic, Technical Report No. 846, Department of Statistics, The Ohio State University, Columbus, OH.

See Also

aple

Examples

```
## Not run:
example(aple)
plt_out <- aple.plot(scale(wheat$yield_detrend, scale=FALSE),
 nb2listw(nbr12, style="W"), cex=0.6)
crossprod(plt_out$Y, plt_out$X)/crossprod(plt_out$X)
lm_obj <- lm(Y ~ X, plt_out)
abline(lm_obj)
abline(v=0, h=0, lty=2)
zz <- summary(influence.measures(lm_obj))
infl <- as.integer(rownames(zz))
points(plt_out$X[infl], plt_out$Y[infl], pch=3, cex=0.6, col="red")
wheat$localAple <- localAple(scale(wheat$yield_detrend, scale=FALSE),
 nb2listw(nbr12, style="W"))
mean(wheat$localAple)
hist(wheat$localAple)
spl <- list("sp.text", coordinates(wheat)[infl,], rep("*", length(infl)))
spplot(wheat, "localAple", sp.layout=spl)

## End(Not run)
```

---

as_dgRMatrix_listw *Interface between Matrix class objects and weights lists*

Description

Interface between Matrix class objects and weights lists

Usage

```
as_dgRMatrix_listw(listw)
as_dsTMatrix_listw(listw)
as_dsCMatrix_I(n)
as_dsCMatrix_IrW(W, rho)
Jacobian_W(W, rho)
```

Arguments

`listw`	a `listw` object created for example by `nb2listw`
`W`	a `dsTMatrix` object created using `as_dsTMatrix_listw` from a symmetric `listw` object
`rho`	spatial regression coefficient
`n`	length of diagonal for identity matrix

Value

Matrix package class objects

Author(s)

Roger Bivand

Examples

```
example(NY_data)
W <- as_dsTMatrix_listw(listw_NY)
I <- as_dsCMatrix_I(dim(W)[1])
W <- as(W, "CsparseMatrix")
rho <- 0.1
c(determinant(I - rho * W, logarithm=TRUE)$modulus)
sum(log(1 - rho * eigenw(listw_NY)))
n <- dim(W)[1]
nW <- - W
nChol <- Cholesky(nW, Imult=8)
.f <- if(package_version(packageDescription("Matrix")$Version) >
           "0.999375-30") 2 else 1
n * log(rho) + (.f * c(determinant(update(nChol, nW, 1/rho))$modulus))
rho <- seq(0.01, 0.1, 0.01)
n * log(rho) + Matrix:::ldetL2up(nChol, nW, 1/rho)
```

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

Description

(Use `example(auckland)` to load the data from shapefile and generate neighbour list on the fly).

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

M77_85 a numeric vector of counts of infant (under 5 years of age) deaths in Auckland, 1977-1985

Und5_81 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.

Examples

```
auckland <- readShapePoly(system.file("etc/shapes/auckland.shp",
 package="spdep")[1])
auckland.nb <- poly2nb(auckland)
```

`autocov_dist` *Distance-weighted autocovariate*

Description

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

Usage

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

Arguments

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

Value

A numeric vector of autocovariate values

Author(s)

Carsten F. Dormann and Roger Bivand

References

Augustin N.H., Mugglestone M.A. and Buckland S.T. (1996) An autologistic model for the spatial distribution of wildlife. *Journal of Applied Ecology*, 33, 339-347; Gumpertz M.L., Graham J.M. and Ristaino J.B. (1997) Autologistic model of spatial pattern of Phytophthora epidemic in bell pepper: effects of soil variables on disease presence. *Journal of Agricultural, Biological and Environmental Statistics*, 2, 131-156.

See Also

`nb2listw`

Examples

```
example(columbus)
xy <- cbind(columbus$X, columbus$Y)
ac1a <- autocov_dist(columbus$CRIME, xy, nbs=10, style="W",
 type="one")
acinva <- autocov_dist(columbus$CRIME, xy, nbs=10, style="W",
 type="inverse")
acinv2a <- autocov_dist(columbus$CRIME, xy, nbs=10, style="W",
 type="inverse.squared")

plot(ac1a ~ columbus$CRIME, pch=16, asp=1)
points(acinva ~ columbus$CRIME, pch=16, col="red")
```

```
points(acinv2a ~ columbus$CRIME, pch=16, col="blue")
abline(0,1)

nb <- dnearneigh(xy, 0, 10)
lw <- nb2listw(nb, style="W")
ac1b <- lag(lw, columbus$CRIME)
all.equal(ac1b, ac1a)

nbd <- nbdists(nb, xy)
gl <- lapply(nbd, function(x) 1/x)
lw <- nb2listw(nb, glist=gl)
acinvb <- lag(lw, columbus$CRIME)
all.equal(acinvb, acinva)

gl2 <- lapply(nbd, function(x) 1/(x^2))
lw <- nb2listw(nb, glist=gl2)
acinv2b <- lag(lw, columbus$CRIME)
all.equal(acinv2b, acinv2a)

glm(CRIME ~ HOVAL + ac1b, data=columbus, family="gaussian")
spautolm(columbus$CRIME ~ HOVAL, data=columbus,
 listw=nb2listw(nb, style="W"))

xy <- SpatialPoints(xy)
acinva <- autocov_dist(columbus$CRIME, xy, nbs=10, style="W",
 type="inverse")
nb <- dnearneigh(xy, 0, 10)
nbd <- nbdists(nb, xy)
gl <- lapply(nbd, function(x) 1/x)
lw <- nb2listw(nb, glist=gl)
acinvb <- lag(lw, columbus$CRIME)
all.equal(acinvb, acinva)
```

`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) ...
```

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

Description

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

Format

A shape polygon object with seven variables:

id The identificator

Name Name of city

Population The population of city

HLCI Health Life Condition Index

ELCI Education Life Condition Index

CLCI Children Life Condition Index

ELCI Economic Life Condition Index

Examples

```
### see example in 'skater' function help
```

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))
## Not run: 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="Matrix",
 control=list(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="Matrix")
summary(gp2)
## End(Not run)
```

---

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, varformula=NULL, studentize = TRUE, data=list())
```

Arguments

`object`	An object of class `"sarlm"` from `errorsarlm()` or `lagsarlm()`.
`varformula`	a formula describing only the potential explanatory variables for the variance (no dependent variable needed). By default the same explanatory variables are taken as in the main regression model
`studentize`	logical. If set to `TRUE` Koenker's studentized version of the test statistic will be used.
`data`	an optional data frame containing the variables in the varformula

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).

It is also technically possible to make heteroskedasticity corrections to standard error estimates by using the "lm.target" component of `sarlm` objects - using functions in the lmtest and sandwich packages.

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.`
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

```
example(columbus)
error.col <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus,
 nb2listw(col.gal.nb))
bptest.sarlm(error.col)
bptest.sarlm(error.col, studentize=FALSE)
## Not run:
if (require(lmtest) && require(sandwich)) {
  coeftest(error.col$lm.target, vcov=vcovHC(error.col$lm.target,
    type="HC0"), df=Inf)
}

## End(Not run)
```

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

```
example(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, legacy=FALSE)
```

Arguments

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

Value

A data frame with columns:

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

probmap

Examples

```
example(auckland)
res <- choynowski(auckland$M77_85, 9*auckland$Und5_81)
resl <- choynowski(auckland$M77_85, 9*auckland$Und5_81, legacy=TRUE)
all.equal(res, resl)
rt <- sum(auckland$M77_85)/sum(9*auckland$Und5_81)
ch_ppois_pmap <- numeric(length(auckland$Und5_81))
side <- c("greater", "less")
for (i in seq(along=ch_ppois_pmap)) {
  ch_ppois_pmap[i] <- poisson.test(auckland$M77_85[i], r=rt,
    T=(9*auckland$Und5_81[i]), alternative=side[(res$type[i]+1)])$p.value
}
all.equal(ch_ppois_pmap, res$pmap)

res1 <- probmap(auckland$M77_85, 9*auckland$Und5_81)
table(abs(res$pmap - res1$pmap) < 0.00001, res$type)
lt005 <- (res$pmap < 0.05) & (res$type)
ge005 <- (res$pmap < 0.05) & (!res$type)
cols <- rep("white", length(lt005))
cols[lt005] <- grey(2/7)
cols[ge005] <- grey(5/7)
plot(auckland, col=cols)
legend("bottomleft", 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 plumbing

DISCBD distance to CBD

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

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

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

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

```
columbus <- readShapePoly(system.file("etc/shapes/columbus.shp",
 package="spdep")[1])
col.gal.nb <- read.gal(system.file("etc/weights/columbus.gal",
 package="spdep")[1])
```

diffnb *Differences between neighbours lists*

Description

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

Usage

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

Arguments

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

Value

A neighbours list with class `nb`

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
example(columbus)
coords <- coordinates(columbus)
rn <- sapply(slot(columbus, "polygons"), function(x) slot(x, "ID"))
knn1 <- knearneigh(coords, 1)
knn2 <- knearneigh(coords, 2)
nb1 <- knn2nb(knn1, row.names=rn)
nb2 <- knn2nb(knn2, row.names=rn)
diffs <- diffnb(nb2, nb1)
plot(columbus, border="grey")
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 longlat = TRUE, by Great Circle distance in kilometers.

Usage

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

Arguments

`x`	matrix of point coordinates or a SpatialPoints object
`d1`	lower distance bound
`d2`	upper distance bound
`row.names`	character vector of region ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))`
`longlat`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself

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

```
example(columbus)
coords <- coordinates(columbus)
rn <- sapply(slot(columbus, "polygons"), function(x) slot(x, "ID"))
k1 <- knn2nb(knearneigh(coords))
all.linked <- max(unlist(nbdists(k1, coords)))
col.nb.0.all <- dnearneigh(coords, 0, all.linked, row.names=rn)
summary(col.nb.0.all, coords)
plot(columbus, border="grey")
```

```
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, longlat=FALSE))
all.linked <- max(unlist(nbdists(llk1, xy, longlat=FALSE)))
ll.nb <- dnearneigh(xy, 0, all.linked, longlat=FALSE)
summary(ll.nb, xy, longlat=TRUE, scale=0.5)
gck1 <- knn2nb(knearneigh(xy, k=1, longlat=TRUE))
all.linked <- max(unlist(nbdists(gck1, xy, longlat=TRUE)))
gc.nb <- dnearneigh(xy, 0, all.linked, longlat=TRUE)
summary(gc.nb, xy, longlat=TRUE, scale=0.5)
plot(ll.nb, xy)
plot(diffnb(ll.nb, gc.nb), xy, add=TRUE, col="red", lty=2)
title(main="Differences between Euclidean and Great Circle neighbours")

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

---

do_ldet *Spatial regression model Jacobian computations*

Description

These functions are made available in the package namespace for other developers, and are not intended for users. They provide a shared infrastructure for setting up data for Jacobian computation, and then for caclulating the Jacobian, either exactly or approximately, in maximum likelihood fitting of spatial regression models. The techniques used are the exact eigenvalue, Cholesky decompositions (Matrix, spam), and LU ones, with Chebyshev and Monte Carlo approximations; moments use the methods due to Martin and Smirnov/Anselin.

Usage

```
do_ldet(coef, env, which=1)
cheb_setup(env, q=5, which=1)
mcdet_setup(env, p=16, m=30, which=1)
eigen_setup(env, which=1)
```

```
spam_setup(env, pivot="MMD", which=1)
spam_update_setup(env, in_coef=0.1, pivot="MMD", which=1)
Matrix_setup(env, Imult, super=as.logical(NA), which=1)
Matrix_J_setup(env, super=FALSE, which=1)
LU_setup(env, which=1)
moments_setup(env, trs=NULL, m, p, type="MC", correct=TRUE, trunc=TRUE, which=1)
```

Arguments

`coef`	spatial coefficient value
`env`	environment containing pre-computed objects, fixed after assignment in setup functions
`which`	default 1; if 2, use second listw object
`q`	Chebyshev approximation order; default in calling spdep functions is 5, here it cannot be missing and does not have a default
`p`	Monte Carlo approximation number of random normal variables; default calling spdep functions is 16, here it cannot be missing and does not have a default
`m`	Monte Carlo approximation number of series terms; default in calling spdep functions is 30, here it cannot be missing and does not have a default; `m` serves the same purpose in the moments method
`pivot`	default "MMD", may also be "RCM" for Cholesky decompisition using spam
`in_coef`	fill-in initiation coefficient value, default 0.1
`Imult`	see `Cholesky`; numeric scalar which defaults to zero. The matrix that is decomposed is A+m*I where m is the value of Imult and I is the identity matrix of order ncol(A). Default in calling spdep functions is 2, here it cannot be missing and does not have a default, but is rescaled for binary weights matrices in proportion to the maximim row sum in those calling functions
`super`	see `Cholesky`; logical scalar indicating is a supernodal decomposition should be created. The alternative is a simplicial decomposition. Default in calling spdep functions is FALSE for "Matrix_J" and `as.logical(NA)` for "Matrix". Setting it to NA leaves the choice to a CHOLMOD-internal heuristic
`trs`	A numeric vector of `m` traces, as from `trW`
`type`	moments trace type, see `trW`
`correct`	default TRUE: use Smirnov correction term, see `trW`
`trunc`	default TRUE: truncate Smirnov correction term, see `trW`

Details

Since environments are containers in the R workspace passed by reference rather than by value, they are useful for passing objects to functions called in numerical optimisation, here for the maximum likelihood estimation of spatial regression models. This technique can save a little time on each function call, balanced against the need to access the objects in the environment inside the function. The environment should contain a `family` string object either "SAR", "CAR" or "SMA" (used in `do_ldet` to choose spatial moving average in `spautolm`, and these specific objects before calling the set-up functions:

eigen Classical Ord eigenvalue computations:

listw A listw spatial weights object
can.sim logical scalar: can the spatial weights be made symmetric by similarity
verbose logical scalar: legacy report print control, for historical reasons only

and assigns to the environment:

eig a vector of eigenvalues
eig.range the search interval for the spatial coefficient
method string: "eigen"

Matrix Sparse matrix pre-computed Cholesky decomposition with fast updating:

listw A listw spatial weights object
can.sim logical scalar: can the spatial weights be made symmetric by similarity

and assigns to the environment:

csrw sparse spatial weights matrix
nW negative sparse spatial weights matrix
pChol a "CHMfactor" from factorising `csrw` with `Cholesky`
nChol a "CHMfactor" from factorising `nW` with `Cholesky`
method string: "Matrix"

Matrix_J Standard Cholesky decomposition without updating:

listw A listw spatial weights object
can.sim logical scalar: can the spatial weights be made symmetric by similarity
n number of spatial objects

and assigns to the environment:

csrw sparse spatial weights matrix
I sparse identity matrix
super the value of the `super` argument
method string: "Matrix_J"

spam Standard Cholesky decomposition without updating:

listw A listw spatial weights object
can.sim logical scalar: can the spatial weights be made symmetric by similarity
n number of spatial objects

and assigns to the environment:

csrw sparse spatial weights matrix
I sparse identity matrix
pivot string — pivot method
method string: "spam"

spam_update Pre-computed Cholesky decomposition with updating:

listw A listw spatial weights object
can.sim logical scalar: can the spatial weights be made symmetric by similarity
n number of spatial objects

and assigns to the environment:

csrw sparse spatial weights matrix
I sparse identity matrix
csrwchol A Cholesky decomposition for updating
method string: "spam"

LU Standard LU decomposition without updating:

listw A listw spatial weights object
n number of spatial objects

and assigns to the environment:

W sparse spatial weights matrix
I sparse identity matrix
method string: "LU"

MC Monte Carlo approximation:

listw A listw spatial weights object

and assigns to the environment:

clx list of Monte Carlo approximation terms
W sparse spatial weights matrix
method string: "MC"

cheb Chebyshev approximation:

listw A listw spatial weights object

and assigns to the environment:

trT vector of Chebyshev approximation terms
W sparse spatial weights matrix
method string: "Chebyshev"

moments moments approximation:

listw A listw spatial weights object
can.sim logical scalar: can the spatial weights be made symmetric by similarity

and assigns to the environment:

trs vector of Chebyshev approximation terms
correct logical scalar: use Smirnov correction term
trunc logical scalar: truncate Smirnov correction term
method string: "moments"

Some set-up functions may also assign `similar` to the environment if the weights were made symmetric by similarity.

Value

`do_ldet` returns the value of the Jacobian for the calculation method recorded in the environment argument; the remaining functions modify the environment in place as a side effect and return nothing.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton, pp. 77–110

See Also

spautolm, lagsarlm, errorsarlm, Cholesky

Examples

```
data(boston)
lw <- nb2listw(boston.soi)
can.sim <- spdep:::can.be.simmed(lw)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("verbose", FALSE, envir=env)
assign("family", "SAR", envir=env)
eigen_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
assign("n", length(boston.soi), envir=env)
Matrix_setup(env, Imult=2, super=FALSE)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
spam_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
```

```
LU_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
cheb_setup(env, q=5)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
set.seed(12345)
mcdet_setup(env, p=16, m=30)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
```

`droplinks` *Drop links in a neighbours list*

Description

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

Usage

```
droplinks(nb, drop, sym=TRUE)
```

Arguments

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

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

is.symmetric.nb

Examples

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

EBest *Global Empirical Bayes estimator*

Description

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

Usage

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

Arguments

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

Details

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

Value

A data frame with two columns:

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

and a `parameters` attribute list with components:

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

Author(s)

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

References

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

See Also

`EBlocal`, `probmap`, `EBImoran.mc`

Examples

```
example(auckland)
res <- EBest(auckland$M77_85, 9*auckland$Und5_81)
attr(res, "parameters")
cols <- grey(6:2/7)
brks <- c(-Inf,2,2.5,3,3.5,Inf)
plot(auckland, col=cols[findInterval(res$estmm*1000, brks, all.inside=TRUE)])
legend("bottomleft", fill=cols, legend=leglabs(brks), bty="n")
title(main="Global moment estimator of infant mortality per 1000 per year")
data(huddersfield)
```

```
res <- EBest(huddersfield$cases, huddersfield$total, family="binomial")
round(res[,1:2],4)*100
```

EBImoran.mc *Permutation test for empirical Bayes index*

Description

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

Usage

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

Arguments

n	a numeric vector of counts of cases the same length as the neighbours list in listw
x	a numeric vector of populations at risk the same length as the neighbours list in listw
listw	a `listw` object created for example by `nb2listw`
nsim	number of permutations
zero.policy	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative	a character string specifying the alternative hypothesis, must be one of "greater" (default), or "less"
spChk	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
return_boot	return an object of class `boot` from the equivalent permutation bootstrap rather than an object of class `htest`

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

```
example(nc.sids)
EBImoran.mc(nc.sids$SID74, nc.sids$BIR74,
 nb2listw(ncCC89_nb, style="B", zero.policy=TRUE), nsim=999, zero.policy=TRUE)
sids.p <- nc.sids$SID74 / nc.sids$BIR74
moran.mc(sids.p, nb2listw(ncCC89_nb, style="B", zero.policy=TRUE),
 nsim=999, zero.policy=TRUE)
```

EBlocal — *Local Empirical Bayes estimator*

Description

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

Usage

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

Arguments

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

Details

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

Value

A data frame with two columns:

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

and a `parameters` attribute list with components:

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

Author(s)

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

References

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

See Also

EBest, probmap

Examples

```
example(auckland)
res <- EBlocal(auckland$M77_85,  9*auckland$Und5_81, auckland.nb)
brks <- c(-Inf,2,2.5,3,3.5,Inf)
cols <- grey(6:2/7)
plot(auckland, col=cols[findInterval(res$est*1000, brks, all.inside=TRUE)])
legend("bottomleft", fill=cols, legend=leglabs(brks), bty="n")
title(main="Local moment estimator of infant mortality per 1000 per year")
```

---

edit.nb *Interactive editing of neighbours lists*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb, plot.nb

Examples

```
## Not run:
data(columbus)
class(polys)
nnb <- edit.nb(col.gal.nb, coords, polys)
example(columbus)
class(columbus)
nnb1 <- edit.nb(col.gal.nb, polys=columbus)

## End(Not run)
```

---

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 λ_i are the eigenvalues of W.

Usage

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

Arguments

listw	a listw object created for example by nb2listw
quiet	default NULL, use global !verbose option value; 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 data set has been converted to shapefile format and placed in the etc/shapes directory. The initial data objects are now stored as a SpatialPolygonsDataFrame object, from which the contiguity neighbour list is recreated. For purposes of record, the original data set is retained.

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 £000

INCOME total personal income £000

names County names

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, originally hosted at: `http://geog-www.sbs.ohio-state.edu/faculty/tiefelsdorf/GeoStat.htm`.

Examples

```
eire <- readShapePoly(system.file("etc/shapes/eire.shp", package="spdep")[1],
  ID="names", proj4string=CRS("+proj=utm +zone=30 +units=km"))
eire.nb <- poly2nb(eire)
#data(eire)

summary(eire$A)
brks <- round(fivenum(eire$A), digits=2)
cols <- rev(heat.colors(4))
plot(eire, col=cols[findInterval(eire$A, brks, all.inside=TRUE)])
title(main="Percentage with blood group A in Eire")
legend(x=c(-50, 70), y=c(6120, 6050), leglabs(brks), fill=cols, bty="n")
plot(eire)
plot(eire.nb, coordinates(eire), add=TRUE)
lA <- lag.listw(nb2listw(eire.nb), eire$A)
summary(lA)
moran.test(eire$A, nb2listw(eire.nb))
geary.test(eire$A, nb2listw(eire.nb))
cor(lA, eire$A)
moran.plot(eire$A, nb2listw(eire.nb),
 labels=eire$names)
A.lm <- lm(A ~ towns + pale, data=eire)
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, col=cols[findInterval(res, brks, all.inside=TRUE)])
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$OWNCONS), digits=2)
cols <- grey(4:1/5)
plot(eire, col=cols[findInterval(eire$OWNCONS, brks, all.inside=TRUE)])
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(eire$OWNCONS, nb2listw(eire.nb))
moran.test(eire$OWNCONS, nb2listw(eire.nb))
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
res <- residuals(e.lm)
brks <- c(min(res),-2,-1,0,1,2,max(res))
cols <- rev(cm.colors(6))
plot(eire, col=cols[findInterval(res, brks, all.inside=TRUE)])
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:length(slot(eire, "polygons"))))
```

---

elect80 *1980 Presidential election results*

Description

A data set for 1980 Presidential election results covering 3,107 US counties using geographical coordinates. In addition, three spatial neighbour objects, `k4` not using Great Circle distances, `dll` using Great Circle distances, and `e80_queen` of Queen contiguities for equivalent County polygons taken from file `co1980p020.tar.gz` on the USGS National Atlas site, and a spatial weights object imported from `elect.ford` - a 4-nearest neighbour non-GC row-standardised object, but with coercion to symmetry.

Usage

```
data(elect80)
```

Format

A SpatialPointsDataFrame with 3107 observations on the following 7 variables.

`FIPS` a factor of county FIPS codes

`long` a numeric vector of longitude values

`lat` a numeric vector of latitude values

`pc_turnout` Votes cast as proportion of population over age 19 eligible to vote

`pc_college` Population with college degrees as proportion of population over age 19 eligible to vote

`pc_homeownership` Homeownership as proportion of population over age 19 eligible to vote

`pc_income` Income per capita of population over age 19 eligible to vote

Source

Pace, R. Kelley and Ronald Barry. 1997. "Quick Computation of Spatial Autoregressive Estimators", in Geographical Analysis; sourced from the data folder in the Spatial Econometrics Toolbox for Matlab, http://www.spatial-econometrics.com/html/jplv7.zip, files `elect.dat` and `elect.ford` (with the final line dropped).

Examples

```
data(elect80)
```

`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 λ is found by `optimize()` first, and β and other parameters by generalized least squares subsequently. With one of the sparse matrix methods, larger numbers of observations can be handled, but the `interval=` argument may need be set when the weights are not row-standardised.

Usage

```
errorsarlm(formula, data=list(), listw, na.action, etype="error",
 method="eigen", quiet=NULL, zero.policy=NULL,
 interval = NULL, tol.solve=1.0e-10, trs=NULL, control=list())
```

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 `options("na.action")`), 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.

etype
: default "error", may be set to "emixed" to include the spatially lagged independent variables added to X; when "emixed", the lagged intercept is dropped for spatial weights style "W", that is row-standardised weights, but otherwise included

method
: "eigen" (default) - the Jacobian is computed as the product of (1 - rho*eigenvalue) using `eigenw`, and "spam" or "Matrix_J" for strictly symmetric weights lists of styles "B" and "C", or made symmetric by similarity (Ord, 1975, Appendix C) if possible for styles "W" and "S", using code from the spam package or Matrix package to calculate the determinant; “Matrix” and “spam_update” provide updating Cholesky decomposition methods; "LU" provides an alternative sparse matrix decomposition approach. In addition, there are "Chebyshev" and Monte Carlo "MC" approximate log-determinant methods; the Smirnov/Anselin (2009) trace approximation is available as "moments".

quiet
: default NULL, use !verbose global option value; if FALSE, reports function values during optimization.

zero.policy
: default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA - causing `errorsarlm()` to terminate with an error

interval
: default is NULL, search interval for autoregressive parameter

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

trs
: default NULL, if given, a vector of powered spatial weights matrix traces output by `trW`; when given, insert the asymptotic analytical values into the numerical Hessian instead of the approximated values; may be used to get around some problems raised when the numerical Hessian is poorly conditioned, generating NaNs in subsequent operations. When using the numerical Hessian to get the standard error of lambda, it is very strongly advised that trs be given, as the

parts of fdHess corresponding to the regression coefficients are badly approximated, affecting the standard error of lambda; the coefficient correlation matrix is unusable

`control` list of extra control arguments - see section below

Details

The asymptotic standard error of λ 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="spam" or "Matrix". 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 λ 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.

Note that the fitted() function for the output object assumes that 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 (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data.

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

`opt` object returned from numerical optimisation

`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 λ
`LMtest`	NULL for this model
`aliased`	if not NULL, details of aliased variables
`LLNullLlm`	Log-likelihood of the null linear model
`Hcov`	Spatial DGP covariance matrix for Hausman test if available
`interval`	line search interval
`fdHess`	finite difference Hessian
`optimHess`	`optim` or `fdHess` used
`insert`	logical; is TRUE, asymptotic values inserted in fdHess where feasible
`timings`	processing timings
`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 λ.

Control arguments

tol.opt: the desired accuracy of the optimization - passed to `optimize()` (default=square root of double precision machine tolerance, a larger root may be used needed, see help(boston) for an example)

returnHcov: default TRUE, return the Vo matrix for a spatial Hausman test

pWOrder: default 250, if returnHcov=TRUE and the method is not “eigen”, pass this order to `powerWeights` as the power series maximum limit

fdHess: default NULL, then set to (method != "eigen") internally; use `fdHess` to compute an approximate Hessian using finite differences when using sparse matrix methods; used to make a coefficient covariance matrix when the number of observations is large; may be turned off to save resources if need be

optimHess: default FALSE, use `fdHess` from **nlme**, if TRUE, use `optim` to calculate Hessian at optimum

LAPACK: default FALSE; logical value passed to `qr` in the SSE log likelihood function

compiled_sse: default FALSE; logical value used in the log likelihood function to choose compiled code for computing SSE

Imult: default 2; used for preparing the Cholesky decompositions for updating in the Jacobian function

super: if NULL (default), set to FALSE to use a simplicial decomposition for the sparse Cholesky decomposition and method “Matrix_J”, set to `as.logical(NA)` for method “Matrix”, if TRUE, use a supernodal decomposition

cheb_q: default 5; highest power of the approximating polynomial for the Chebyshev approximation

MC_p: default 16; number of random variates

MC_m: default 30; number of products of random variates matrix and spatial weights matrix

spamPivot: default "MMD", alternative "RCM"

in_coef default 0.1, coefficient value for initial Cholesky decomposition in "spam_update"

type default "MC", used with method "moments"; alternatives "mult" and "moments", for use if `trs` is missing, `trW`

correct default TRUE, used with method "moments" to compute the Smirnov/Anselin correction term

trunc default TRUE, used with method "moments" to truncate the Smirnov/Anselin correction term

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; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York; LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton.

See Also

`lm`, `lagsarlm`, `similar.listw`, `predict.sarlm`, `residuals.sarlm`, `do_ldet`

Examples

```
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", quiet=FALSE)
summary(COL.errW.eig, correlation=TRUE)
COL.errB.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="B"), method="eigen", quiet=FALSE)
summary(COL.errB.eig, correlation=TRUE)
W <- as(as_dgRMatrix_listw(nb2listw(COL.nb)), "CsparseMatrix")
trMatc <- trW(W, type="mult")
COL.errW.M <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix", quiet=FALSE, trs=trMatc)
summary(COL.errW.M, 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)
```

```
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen"))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", control=list(LAPACK=TRUE)))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", control=list(compiled_sse=TRUE)))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix_J", control=list(super=TRUE)))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix_J", control=list(super=FALSE)))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix_J", control=list(super=as.logical(NA))))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix", control=list(super=TRUE)))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix", control=list(super=FALSE)))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="Matrix", control=list(super=as.logical(NA))))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="spam", control=list(spamPivot="MMD")))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="spam", control=list(spamPivot="RCM")))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="spam_update", control=list(spamPivot="MMD")))
system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="spam_update", control=list(spamPivot="RCM")))
COL.merrW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", etype="emixed")
summary(COL.merrW.eig, correlation=TRUE)
```

`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=NULL)
```

Arguments

x	a numeric vector the same length as the neighbours list in listw
listw	a `listw` object created for example by `nb2listw`
n	number of zones
n1	n - 1
S0	global sum of weights
zero.policy	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

Value

a list with

C	Geary's C
K	sample kurtosis of x

Author(s)

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

References

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

See Also

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

Examples

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

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

Description

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

Usage

```
geary.mc(x, listw, nsim, zero.policy=NULL, alternative="less",
 spChk=NULL, adjust.n=TRUE, return_boot=FALSE)
```

Arguments

- `x` — a numeric vector the same length as the neighbours list in listw
- `listw` — a `listw` object created for example by `nb2listw`
- `nsim` — number of permutations
- `zero.policy` — default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
- `alternative` — a character string specifying the alternative hypothesis, must be one of "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()`
- `adjust.n` — default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
- `return_boot` — return an object of class `boot` from the equivalent permutation bootstrap rather than an object of class `htest`

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

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

Description

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

Usage

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

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`
`randomisation`	variance of I calculated under the assumption of randomisation, if FALSE normality
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`adjust.n`	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

Value

A list with class htest containing the following components:

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

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 21, Cliff, A. D., Ord, J. K. 1973 Spatial Autocorrelation, Pion, pp. 15-16, 21.

See Also

geary, geary.mc, listw2U

Examples

```
data(oldcol)
geary.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"))
geary.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
colold.lags <- nblag(COL.nb, 3)
geary.test(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"))
geary.test(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"), alternative="greater")
```

```
print(is.symmetric.nb(COL.nb))
coords.OLD <- cbind(COL.OLD$X, COL.OLD$Y)
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
geary.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"))
geary.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"),
 randomisation=FALSE)
cat("Note non-symmetric weights matrix - use listw2U()\n")
geary.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")))
geary.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")), randomisation=FALSE)
```

---

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 *Global G test for spatial autocorrelation*

Description

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

Usage

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

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`; if a sequence of distance bands is to be used, it is recommended that the weights style be binary (one of `c("B", "C", "U")`).
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`adjust.n`	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

Value

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

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

Author(s)

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

References

Getis. A, Ord, J. K. 1992 The analysis of spatial association by use of distance statistics, *Geographical Analysis*, 24, p. 195; see also Getis. A, Ord, J. K. 1993 Erratum, *Geographical Analysis*, 25, p. 276.

See Also

localG

Examples

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

GMerrorsar — *Spatial simultaneous autoregressive error model estimation by GMM*

Description

An implementation of Kelejian and Prucha's generalised moments estimator for the autoregressive parameter in a spatial model.

Usage

```
GMerrorsar(formula, data = list(), listw, na.action = na.fail,
 zero.policy = NULL, return_LL = FALSE, method="nlminb",
 control = list(), pars, verbose=NULL, sparse_method="Matrix",
 returnHcov=FALSE, pWOrder=250, tol.Hcov=1.0e-10)
## S3 method for class 'gmsar'
summary(object, correlation = FALSE, Hausman=FALSE, ...)
GMargminImage(obj, lambdaseq, s2seq)
```

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.

`zero.policy` default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `GMerrorsar()` to terminate with an error

`return_LL` default FALSE, if TRUE, try to calculate the log likelihood of the function for the fitted model values — see details

`method` default `"nlminb"`, or optionally a method passed to `optim` to use an alternative optimizer

`control` A list of control parameters. See details in `optim` or `nlminb`.

`pars` starting values for λ and σ^2 for GMM optimisation, if missing (default), approximated from initial OLS model as the autocorrelation coefficient corrected for weights style and model sigma squared

`verbose` default NULL, use global option value; if TRUE, reports function values during optimization.

`sparse_method` default "Matrix", can also be "spam" to use spam package objects for finding the Jacobian

`returnHcov` default FALSE, return the Vo matrix for a spatial Hausman test

`tol.Hcov` the tolerance for computing the Vo matrix (default=1.0e-10)

`pWOrder` default 250, if returnHcov=TRUE, pass this order to `powerWeights` as the power series maximum limit

`object, obj` `gmsar` object from `GMerrorsar`

`correlation` logical; (default=FALSE), TRUE not available

`Hausman` if TRUE, the results of the Hausman test for error models are reported

`...` `summary` arguments passed through

`lambdaseq` if given, an increasing sequence of lambda values for gridding

`s2seq` if given, an increasing sequence of sigma squared values for gridding

Details

When the control list is set with care, the function will converge to values close to the ML estimator without requiring computation of the Jacobian, the most resource-intensive part of ML estimation. For moderately sized data sets with hundreds of observations, but not many thousands, the Jacobian is computed once to give the likelihood of the fitted model, allowing a test against the model with no spatial dependence.

Note that the fitted() function for the output object assumes that 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 (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data.

The `GMargminImage` may be used to visualize the shape of the surface of the argmin function used to find lambda.

Value

A list object of class `gmsar`

`type`	"ERROR"
`lambda`	simultaneous autoregressive error coefficient
`coefficients`	GMM coefficient estimates
`rest.se`	GMM coefficient standard errors
`s2`	GMM residual variance
`SSE`	sum of squared GMM errors
`parameters`	number of parameters estimated
`lm.model`	the `lm` object returned when estimating for $\lambda = 0$
`call`	the call used to create this object
`residuals`	GMM residuals
`lm.target`	the `lm` object returned for the GMM fit
`fitted.values`	Difference between residuals and response variable
`formula`	model formula
`aliased`	if not NULL, details of aliased variables
`zero.policy`	zero.policy for this model
`LL`	log likelihood value at computed optimum
`vv`	list of internal bigG and litg components for testing optimisation surface
`optres`	object returned by optimizer
`pars`	start parameter values for optimisation
`Hcov`	Spatial DGP covariance matrix for Hausman test if available
`na.action`	(possibly) named vector of excluded or omitted observations if non-default na.action argument used

Author(s)

Luc Anselin and Roger Bivand

References

Kelejian, H. H., and Prucha, I. R., 1999. A Generalized Moments Estimator for the Autoregressive Parameter in a Spatial Model. International Economic Review, 40, pp. 509–533; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York.

See Also

`optim`, `nlminb`, `errorsarlm`

Examples

```
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen")
summary(COL.errW.eig, Hausman=TRUE)
COL.errW.GM <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), returnHcov=TRUE)
summary(COL.errW.GM, Hausman=TRUE)
aa <- GMargminImage(COL.errW.GM)
levs <- quantile(aa$z, seq(0, 1, 1/12))
image(aa, breaks=levs, xlab="lambda", ylab="s2")
points(COL.errW.GM$lambda, COL.errW.GM$s2, pch=3, lwd=2)
contour(aa, levels=signif(levs, 4), add=TRUE)
COL.errW.GM1 <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"))
summary(COL.errW.GM1)
example(NY_data)
esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, family="SAR", method="full")
summary(esar1f)
esar1gm <- GMerrorsar(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY)
summary(esar1gm)
esar1gm1 <- GMerrorsar(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, method="Nelder-Mead")
summary(esar1gm1)
```

`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 <nikko@hailmail.net>

See Also

plot.nb

Examples

```
example(columbus)
coords <- coordinates(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)
```

---

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, nnmult=3)
relativeneigh(coords, nnmult=3)

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
`nnmult`	scaling factor for memory allocation, default 3; if higher values are required, the function will exit with an error; example below thanks to Dan Putler
`tri.nb`	a neighbor list created from tri2nb
`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

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

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

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

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 <nikko@hailmail.net>

References

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

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

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

See Also

knearneigh, dnearneigh, knn2nb

Examples

```
example(columbus)
coords <- coordinates(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)
plot(columbus, border="grey")
plot(col.tri.nb,coords,add=TRUE)
title(main="Delaunay Triangulation")
plot(columbus, border="grey")
plot(col.gab.nb, coords, add=TRUE)
title(main="Gabriel Graph")
plot(columbus, border="grey")
plot(col.rel.nb, coords, add=TRUE)
title(main="Relative Neighbor Graph")
plot(columbus, border="grey")
plot(col.soi.nb, coords, add=TRUE)
title(main="Sphere of Influence Graph")
par(mfrow=c(1,1))
dx <- rep(0.25*0:4,5)
dy <- c(rep(0,5),rep(0.25,5),rep(0.5,5), rep(0.75,5),rep(1,5))
m <- cbind(c(dx, dx, 3+dx, 3+dx), c(dy, 3+dy, dy, 3+dy))
try(res <- gabrielneigh(m))
res <- gabrielneigh(m, nnmult=4)
summary(graph2nb(res))
```

---

gstsls — *Spatial simultaneous autoregressive SAC model estimation by GMM*

Description

An implementation of Kelejian and Prucha's generalised moments estimator for the autoregressive parameter in a spatial model with a spatially lagged dependent variable.

Usage

```
gstsls(formula, data = list(), listw, listw2 = NULL, na.action = na.fail,
    zero.policy = NULL, pars, control = list(), verbose = NULL, method = "nlminb",
    robust = FALSE, legacy = FALSE, W2X = TRUE )
```

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`
`listw2`	a `listw` object created for example by `nb2listw`, if not given, set to the same spatial weights as the listw argument
`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.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `GMerrorsar()` to terminate with an error
`pars`	starting values for λ and σ^2 for GMM optimisation, if missing (default), approximated from initial 2sls model as the autocorrelation coefficient corrected for weights style and model sigma squared
`control`	A list of control parameters. See details in optim or nlminb
`verbose`	default NULL, use global option value; if TRUE, reports function values during optimization.
`method`	default nlminb, or optionally a method passed to optim to use an alternative optimizer
`robust`	see `stsls`
`legacy`	see `stsls`
`W2X`	see `stsls`

Details

When the control list is set with care, the function will converge to values close to the ML estimator without requiring computation of the Jacobian, the most resource-intensive part of ML estimation.

Value

A list object of class `gmsar`

`lambda`	simultaneous autoregressive error coefficient

`coefficients`	GMM coefficient estimates (including the spatial autocorrelation coefficient)
`rest.se`	GMM coefficient standard errors
`s2`	GMM residual variance
`SSE`	sum of squared GMM errors
`parameters`	number of parameters estimated
`lm.model`	NULL
`call`	the call used to create this object
`residuals`	GMM residuals
`lm.target`	NULL
`fitted.values`	Difference between residuals and response variable
`formula`	model formula
`aliased`	NULL
`zero.policy`	zero.policy for this model
`LL`	NULL
`vv`	list of internal bigG and litg components for testing optimisation surface
`optres`	object returned by optimizer
`pars`	start parameter values for optimisation
`Hcov`	NULL
`na.action`	(possibly) named vector of excluded or omitted observations if non-default na.action argument used

Author(s)

Gianfranco Piras and Roger Bivand

References

Kelejian, H. H., and Prucha, I. R., 1999. A Generalized Moments Estimator for the Autoregressive Parameter in a Spatial Model. International Economic Review, 40, pp. 509–533; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York.

See Also

`optim`, `nlminb`, `GMerrorsar`, `GMargminImage`

Examples

```
data(oldcol)
COL.errW.GM <- gstsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb, style="W"))
summary(COL.errW.GM)
aa <- GMargminImage(COL.errW.GM)
levs <- quantile(aa$z, seq(0, 1, 1/12))
image(aa, breaks=levs, xlab="lambda", ylab="s2")
points(COL.errW.GM$lambda, COL.errW.GM$s2, pch=3, lwd=2)
contour(aa, levels=signif(levs, 4), add=TRUE)
```

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 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()
```

house — *Lucas county OH housing*

Description

Data on 25,357 single family homes sold in Lucas County, Ohio, 1993-1998 from the county auditor, together with an `nb` neighbour object constructed as a sphere of influence graph from projected coordinates.

Usage

```
data(house)
```

Format

The format is: Formal class 'SpatialPointsDataFrame' [package "sp"] with 5 slots. The data slot is a data frame with 25357 observations on the following 24 variables.

- `price` a numeric vector
- `yrbuilt` a numeric vector
- `stories` a factor with levels `one` `bilevel` `multilvl` `one+half` `two` `two+half` `three`
- `TLA` a numeric vector
- `wall` a factor with levels `stucdrvt` `ccbtile` `metlvnyl` `brick` `stone` `wood` `partbrk`
- `beds` a numeric vector
- `baths` a numeric vector
- `halfbaths` a numeric vector
- `frontage` a numeric vector
- `depth` a numeric vector
- `garage` a factor with levels `no garage` `basement` `attached` `detached` `carport`
- `garagesqft` a numeric vector
- `rooms` a numeric vector
- `lotsize` a numeric vector
- `sdate` a numeric vector
- `avalue` a numeric vector
- `s1993` a numeric vector
- `s1994` a numeric vector
- `s1995` a numeric vector
- `s1996` a numeric vector
- `s1997` a numeric vector
- `s1998` a numeric vector
- `syear` a factor with levels `1993` `1994` `1995` `1996` `1997` `1998`
- `age` a numeric vector

Its projection is `CRS(+init=epsg:2834)`, the Ohio North State Plane.

Source

Dataset included in the Spatial Econometrics toolbox for Matlab, `http://www.spatial-econometrics.com/html/jplv7.zip`.

Examples

```
## Not run:
house <- read.table("house.dat", header=FALSE)
names(house) <- c("price", "yrbuilt", "stories", "TLA", "wall", "beds",
  "baths", "halfbaths", "frontage", "depth", "garage", "garagesqft", "rooms",
  "lotsize", "sdate", "avalue", "long", "lat", "s1993", "s1994", "s1995",
  "s1996", "s1997", "s1998")
house$syear <- 1992 + house$s1993 + 2*house$s1994 + 3*house$s1995 +
4*house$s1996 + 5*house$s1997 + 6*house$s1998
house$syear <- factor(house$syear)
house$age <- (1999 - house$yrbuilt)/100
house$stories <- factor(house$stories, levels=1:7, labels=c("one",
 "bilevel", "multilvl", "one+half", "two", "two+half", "three"))
house$wall <- factor(house$wall, levels=1:7, labels=c("stucdrvt",
 "ccbtile", "metlvnyl", "brick", "stone", "wood", "partbrk"))
house$garage <- factor(house$garage, levels=0:4, labels=c("no garage",
 "basement", "attached", "detached", "carport"))
library(sp)
coordinates(house) <- c("long", "lat")
proj4string(house) <- CRS("+proj=longlat")
library(rgdal)
house <- spTransform(house, CRS("+init=epsg:2834"))
library(spdep)
LO_nb <- graph2nb(soi.graph(tri2nb(coordinates(house)), coordinates(house)))
W <- as(as_dgRMatrix_listw(nb2listw(LO_nb)), "CsparseMatrix")
trMat <- trW(W, type="mult")

## End(Not run)
data(house)
## maybe str(house) ; plot(house) ...
```

---

huddersfield *Prevalence of respiratory symptoms*

Description

Prevalence of respiratory symptoms in 71 school catchment areas in Huddersfield, Northern England

Usage

```
data(huddersfield)
```

Format

A data frame with 71 observations on the following 2 variables.

`cases` Prevalence of at least mild conditions

`total` Number of questionnaires returned

Source

Martuzzi M, Elliott P (1996) Empirical Bayes estimation of small area prevalence of non-rare conditions, Statistics in Medicine 15, 1867–1873, pp. 1870–1871.

Examples

```
data(huddersfield)
str(huddersfield)
```

`impacts` *Impacts in spatial lag models*

Description

The calculation of impacts for spatial lag and spatial Durbin models is needed in order to interpret the regression coefficients correctly, because of the spillovers between the terms in these data generation processes (unlike the spatial error model).

Usage

```
## S3 method for class 'sarlm'
impacts(obj, ..., tr, R = NULL, listw = NULL, useHESS = NULL, tol = 1e-06, empirica
## S3 method for class 'stsls'
impacts(obj, ..., tr, R = NULL, listw = NULL, tol = 1e-06, empirical = FALSE, Q=NUI
## S3 method for class 'lagImpact'
plot(x, ..., choice="direct", trace=FALSE, density=TRUE)
## S3 method for class 'lagImpact'
print(x, ..., reportQ=NULL)
## S3 method for class 'lagImpact'
summary(object, ..., zstats=FALSE, short=FALSE, reportQ=NULL)
## S3 method for class 'lagImpact'
HPDinterval(obj, prob = 0.95, ..., choice="direct")
```

Arguments

`obj`	A sarlm spatial regression object created by `lagsarlm`; in `HPDinterval.lagImpact`, a lagImpact object
`...`	Arguments passed through to methods in the **coda** package
`tr`	A vector of traces of powers of the spatial weights matrix created using `trW`, for approximate impact measures; if not given, `listw` must be given for exact measures (for small to moderate spatial weights matrices); the traces must be for the same spatial weights as were used in fitting the spatial regression
`R`	If given, simulations are used to compute distributions for the impact measures, returned as `mcmc` objects
`listw`	If `tr` is not given, a spatial weights object as created by `nb2listw`; they must be the same spatial weights as were used in fitting the spatial regression

`useHESS`	Use the Hessian approximation (if available) even if the asymptotic coefficient covariance matrix is available; used for comparing methods
`tol`	Argument passed to `mvrnorm`: tolerance (relative to largest variance) for numerical lack of positive-definiteness in the coefficient covariance matrix
`empirical`	Argument passed to `mvrnorm` (default FALSE): if true, the coefficients and their covariance matrix specify the empirical not population mean and covariance matrix
`Q`	default NULL, else an integer number of cumulative power series impacts to calculate if `tr` is given
`reportQ`	default NULL; if TRUE and `Q` given as an argument to `impacts`, report impact components
`x, object`	lagImpact objects created by `impacts` methods
`zstats`	default FALSE, if TRUE, also return z-values and p-values for the impacts based on the simulations
`short`	default FALSE, if TRUE passed to the print summary method to omit printing of the mcmc summaries
`choice`	One of three impacts: direct, indirect, or total
`trace`	Argument passed to `plot.mcmc`: plot trace plots
`density`	Argument passed to `plot.mcmc`: plot density plots
`prob`	Argument passed to `HPDinterval.mcmc`: a numeric scalar in the interval (0,1) giving the target probability content of the intervals

Details

If called without `R` being set, the method returns the direct, indirect and total impacts for the variables in the model, for the variables themselves in tha spatial lag model case, for the variables and their spatial lags in the spatial Durbin (mixed) model case. The spatial lag impact measures are computed using eq. 2.46 (LeSage and Pace, 2009, p. 38), either using the exact dense matrix (when `listw` is given), or traces of powers of the weights matrix (when `tr` is given). When the traces are created by powering sparse matrices, the exact and the trace methods should give very similar results, unless the number of powers used is very small.

If `R` is given, simulations will be used to create distributions for the impact measures, provided that the fitted model object contains a coefficient covariance matrix. At present, this is only the case for the default eigenvalue method in `lagsarlm`, which provides an analytical covariance matrix, but alternatives will be provided for sparse matrix methods in the near future.

The simulations are stored as `mcmc` objects as defined in the **coda** package. The simulated values of the coefficients are checked to see that the spatial coefficient remains within its valid interval — draws outside the interval are discarded.

When `Q` and `tr` are given, addition impact component results are provided for each step in the traces of powers of the weights matrix up to and including the `Q`'th power. This increases computing time because the output object is substantially increased in size in proportion to the size of `Q`.

Value

An object of class lagImpact.

If no simulation is carried out, the object returned is a list with:

`direct` numeric vector

`indirect` numeric vector

`total` numeric vector

and a matching `Qres` list attribute if `Q` was given.

If simulation is carried out, the object returned is a list with:

`res` a list with three components as for the non-simulation case, with a matching `Qres` list attribute if `Q` was given

`sres` a list with three `mcmc` matrices, for the direct, indirect and total impacts with a matching `Qmcmc` list attribute if `Q` was given

Author(s)

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

References

LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton, pp. 33–42, 114–115; LeSage J and MM Fischer (2008) Spatial growth regressions: model specification, estimation and interpretation. Spatial Economic Analysis 3 (3), pp. 275–304.

See Also

`trW`, `lagsarlm`, `nb2listw`, `mvrnorm`, `plot.mcmc`, `summary.mcmc`, `HPDinterval`

Examples

```
example(columbus)
listw <- nb2listw(col.gal.nb)
lobj <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw)
summary(lobj)
mobj <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed")
summary(mobj)
W <- as(as_dgRMatrix_listw(listw), "CsparseMatrix")
trMatc <- trW(W, type="mult")
trMC <- trW(W, type="MC")
impacts(lobj, listw=listw)
impacts(lobj, tr=trMatc)
impacts(lobj, tr=trMC)
lobj1 <- stsls(CRIME ~ INC + HOVAL, columbus, listw)
loobj1 <- impacts(lobj1, tr=trMatc, R=200)
summary(loobj1, zstats=TRUE, short=TRUE)
lobj1r <- stsls(CRIME ~ INC + HOVAL, columbus, listw, robust=TRUE)
loobj1r <- impacts(lobj1r, tr=trMatc, R=200)
summary(loobj1r, zstats=TRUE, short=TRUE)
```

```
lobjIQ5 <- impacts(lobj, tr=trMatc, R=200, Q=5)
summary(lobjIQ5, zstats=TRUE, short=TRUE)
summary(lobjIQ5, zstats=TRUE, short=TRUE, reportQ=TRUE)
impacts(mobj, listw=listw)
impacts(mobj, tr=trMatc)
impacts(mobj, tr=trMC)
summary(impacts(mobj, tr=trMatc, R=200), zstats=TRUE)
## Not run:
mobj1 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="Matrix", fdHess=TRUE)
summary(mobj1)
summary(impacts(mobj1, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
summary(impacts(mobj, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
mobj2 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="Matrix", fdHess=TRUE, optimHess=TRUE)
summary(impacts(mobj2, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
\dontrun{
mobj3 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="spam", fdHess=TRUE)
summary(impacts(mobj3, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
}
data(boston)
Wb <- as(as_dgRMatrix_listw(nb2listw(boston.soi)), "CsparseMatrix")
trMatb <- trW(Wb, type="mult")
gp2mMi <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed", method="Matrix",
fdHess=TRUE, trs=trMatb)
summary(gp2mMi)
summary(impacts(gp2mMi, tr=trMatb, R=1000), zstats=TRUE, short=TRUE)
data(house)
lw <- nb2listw(LO_nb)
form <- formula(log(price) ~ age + I(age^2) + I(age^3) + log(lotsize) +
   rooms + log(TLA) + beds + syear)
lobj <- lagsarlm(form, house, lw, method="Matrix",
 fdHess=TRUE, trs=trMat)
summary(lobj)
loobj <- impacts(lobj, tr=trMat, R=1000)
summary(loobj, zstats=TRUE, short=TRUE)
lobj1 <- stsls(form, house, lw)
loobj1 <- impacts(lobj1, tr=trMat, R=1000)
summary(loobj1, zstats=TRUE, short=TRUE)
mobj <- lagsarlm(form, house, lw, type="mixed",
 method="Matrix", fdHess=TRUE, trs=trMat)
summary(mobj)
moobj <- impacts(mobj, tr=trMat, R=1000)
summary(moobj, zstats=TRUE, short=TRUE)

## End(Not run)
```

---

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

```
example(columbus)
coords <- coordinates(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", method="solve",
 feasible=NULL)
invIrW(listw, rho, method="solve", feasible=NULL)
powerWeights(W, rho, order=250, X, tol=.Machine$double.eps^(3/5))
```

Arguments

`neighbours`	an object of class `nb`
`rho`	autoregressive parameter
`glist`	list of general weights corresponding to neighbours
`style`	`style` can take values W, B, C, and S
`method`	default `solve`, can also take value `chol`
`feasible`	if NULL, the given value of rho is checked to see if it lies within its feasible range, if TRUE, the test is not conducted
`listw`	a `listw` object from for example `nb2listw`
`W`	A spatial weights matrix (either a dense matrix or a CsparseMatrix)
`order`	Power series maximum limit
`X`	A numerical matrix
`tol`	Tolerance for convergence of power series

Details

The `invIrW` 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

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

. With method="chol", Cholesky decomposition is used, thanks to contributed code by Markus Reder and Werner Mueller.

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

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

from

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

, which can be done by storing only sparse V and several matrices of the same dimensions as X. This makes it possible to handle larger spatial weights matrices.

Value

An nxn matrix with a "call" attribute; the `powerWeights` function returns a matrix of the same dimensions as X which has been multipled by the power series equivalent of the dense matrix

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

.

Author(s)

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

References

Tiefelsdorf, M., Griffith, D. A., Boots, B. 1999 A variance-stabilizing coding scheme for spatial link matrices, Environment and Planning A, 31, pp. 165-180; Tiefelsdorf, M. 2000 Modelling spatial processes, Lecture notes in earth sciences, Springer, p. 76; Haining, R. 1990 Spatial data analysis in the social and environmental sciences, Cambridge University Press, p. 117; Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 152; Reder, M. and Mueller, W. (2007) An Improvement of the invIrM Routine of the Geostatistical R-package spdep by Cholesky Inversion, Statistical Projects, LV No: 238.205, SS 2006, Department of Applied Statistics, Johannes Kepler University, Linz

See Also

nb2listw

Examples

```
nb7rt <- cell2nb(7, 7, torus=TRUE)
set.seed(1)
x <- matrix(rnorm(500*length(nb7rt)), nrow=length(nb7rt))
res0 <- apply(invIrM(nb7rt, rho=0.0, method="chol",
 feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res2 <- apply(invIrM(nb7rt, rho=0.2, method="chol",
 feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res4 <- apply(invIrM(nb7rt, rho=0.4, method="chol",
 feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res6 <- apply(invIrM(nb7rt, rho=0.6, method="chol",
 feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res8 <- apply(invIrM(nb7rt, rho=0.8, method="chol",
 feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res9 <- apply(invIrM(nb7rt, rho=0.9, method="chol",
 feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
plot(density(res9), col="red", xlim=c(-0.01, max(density(res9)$x)),
  ylim=range(density(res0)$y),
  xlab="estimated variance of the mean",
  main=expression(paste("Effects of spatial autocorrelation for different ",
    rho, " values")))
lines(density(res0), col="black")
lines(density(res2), col="brown")
lines(density(res4), col="green")
lines(density(res6), col="orange")
lines(density(res8), col="pink")
legend(c(-0.02, 0.01), c(7, 25),
 legend=c("0.0", "0.2", "0.4", "0.6", "0.8", "0.9"),
 col=c("black", "brown", "green", "orange", "pink", "red"), lty=1, bty="n")
## Not run:
x <- matrix(rnorm(length(nb7rt)), ncol=1)
system.time(e <- invIrM(nb7rt, rho=0.9, method="chol", feasible=TRUE) %*% x)
system.time(e <- invIrM(nb7rt, rho=0.9, method="chol", feasible=NULL) %*% x)
system.time(e <- invIrM(nb7rt, rho=0.9, method="solve", feasible=TRUE) %*% x)
system.time(e <- invIrM(nb7rt, rho=0.9, method="solve", feasible=NULL) %*% x)
W <- as(as_dgRMatrix_listw(nb2listw(nb7rt)), "CsparseMatrix")
```

```
system.time(ee <- powerWeights(W, rho=0.9, X=x))
all.equal(e, as(ee, "matrix"), check.attributes=FALSE)
nb60rt <- cell2nb(60, 60, torus=TRUE)
W <- as(as_dgRMatrix_listw(nb2listw(nb60rt)), "CsparseMatrix")
set.seed(1)
x <- matrix(rnorm(dim(W)[1]), ncol=1)
system.time(ee <- powerWeights(W, rho=0.3, X=x))
str(as(ee, "matrix"))
obj <- errorsarlm(as(ee, "matrix")[,1] ~ 1, listw=nb2listw(nb60rt), method="Matrix")
coefficients(obj)

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

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

Value

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

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`read.gal`

Examples

```
example(columbus)
coords <- coordinates(columbus)
ind <- sapply(slot(columbus, "polygons"), function(x) slot(x, "ID"))
print(is.symmetric.nb(col.gal.nb, verbose=TRUE, force=TRUE))
k4 <- knn2nb(knearneigh(coords, k=4), row.names=ind)
k4 <- sym.attr.nb(k4)
print(is.symmetric.nb(k4))
k4.sym <- make.sym.nb(k4)
print(is.symmetric.nb(k4.sym))
```

---

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

Description

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

Usage

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

Arguments

`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, adjust.n=TRUE)
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
`adjust.n`	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted consistently (up to and including spdep 0.3-28 the adjustment was inconsistent - thanks to Tomoki NAKAYA for a careful bug report)
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`x`	object to be printed
`...`	arguments to be passed through for printing

Value

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

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 20; Upton, G., Fingleton, B. 1985 Spatial data analysis by example: point pattern and 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"))
## Not run:
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")

## End(Not run)
```

---

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

Description

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

Usage

```
joincount.test(fx, listw, zero.policy=NULL, alternative="greater", spChk=NULL,
 adjust.n=TRUE)
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`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
`adjust.n`	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted consistently (up to and including spdep 0.3-28 the adjustment was inconsistent - thanks to Tomoki NAKAYA for a careful bug report)
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`x`	object to be printed
`...`	arguments to be passed through for printing

Value

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

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

Note

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

Author(s)

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

References

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

See Also

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

Examples

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

`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 longlat = TRUE, Great Circle distances are used.

Usage

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

Arguments

`x`	matrix of region point coordinates or a SpatialPoints object
`k`	number of nearest neighbours to be returned
`longlat`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself
`RANN`	logical value, if the RANN package is available, use for finding k nearest neighbours when longlat is FALSE

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

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

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

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

```
example(columbus)
coords <- coordinates(columbus)
col.knn <- knearneigh(coords, k=4)
plot(columbus, border="grey")
plot(knn2nb(col.knn), coords, add=TRUE)
title(main="K nearest neighbours, k = 4")
```

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

Description

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

Usage

```
lag.listw(x, var, zero.policy=NULL, NAOK=FALSE, ...)
```

Arguments

`x`	a `listw` object created for example by `nb2listw`
`var`	a numeric vector the same length as the neighbours list in listw
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`NAOK`	If 'FALSE', the presence of 'NA' values is regarded as an error; if 'TRUE' then any 'NA' or 'NaN' or 'Inf' values in var are represented as an NA lagged value.
`...`	additional arguments

Value

a numeric vector the same length as var

Author(s)

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

See Also

`nb2listw`

Examples

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

lagmess *Matrix exponential spatial lag model*

Description

The function fits a matrix exponential spatial lag model, using `optim` to find the value of `alpha`, the spatial coefficient.

Usage

```
lagmess(formula, data = list(), listw, zero.policy = NULL, na.action = na.fail, q =
## S3 method for class 'lagmess'
summary(object, ...)
## S3 method for class 'lagmess'
print(x, ...)
## S3 method for class 'summary.lagmess'
print(x, digits = max(5, .Options$digits - 3),
    signif.stars = FALSE, ...)
## S3 method for class 'lagmess'
residuals(object, ...)
## S3 method for class 'lagmess'
deviance(object, ...)
## S3 method for class 'lagmess'
coef(object, ...)
## S3 method for class 'lagmess'
fitted(object, ...)
## S3 method for class 'lagmess'
logLik(object, ...)
```

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`
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA - causing `lagmess()` to terminate with an error
`na.action`	a function (default `options("na.action")`), 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.

q	default 10; number of powers of the spatial weights to use
start	starting value for numerical optimization, should be a small negative number
control	control parameters passed to optim
method	default BFGS, method passed to optim
verbose	default NULL, use global option value; if TRUE report function values during optimization
x,object	Objects of classes lagmess or summary.lagmess to be passed to methods
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

Details

The underlying spatial lag model:

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

where ρ is the spatial parameter may be fitted by maximum likelihood. In that case, the log likelihood function includes the logartithm of cumbersome Jacobian term $|I - \rho W|$. If we rewrite the model as:

$$Sy = X\beta + \varepsilon$$

we see that in the ML case $Sy = (I - \rho W)y$. If W is row-stochastic, S may be expressed as a linear combination of row-stochastic matrices. By pre-computing the matrix $[yWy, W^2 y, ..., W^{q-1} y]$, the term $Sy(\alpha)$ can readily be found by numerical optimization using the matrix exponential approach. α and ρ are related as $\rho = 1 - \exp \alpha$, conditional on the number of matrix power terms taken q.

Value

The function returns an object of class lagmess with components:

lmobj	the lm object returned after fitting alpha
alpha	the spatial coefficient
alphase	the standard error of the spatial coefficient using the numerical Hessian
rho	the value of rho implied by alpha
bestmess	the object returned by optim
q	the number of powers of the spatial weights used
start	the starting value for numerical optimization used
na.action	(possibly) named vector of excluded or omitted observations if non-default na.action argument used
nullLL	the log likelihood of the aspatial model for the same data

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> and Eric Blankmeyer

References

J. P. LeSage and R. K. Pace (2007) A matrix exponential specification. Journal of Econometrics, 140, 190-214; J. P. LeSage and R. K. Pace (2009) Introduction to Spatial Econometrics. CRC Press, Chapter 9.

See Also

lagsarlm, optim

Examples

```
data(baltimore)
baltimore$AGE <- ifelse(baltimore$AGE < 1, 1, baltimore$AGE)
lw <- nb2listw(knn2nb(knearneigh(cbind(baltimore$X, baltimore$Y), k=7)))
obj1 <- lm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT) + lag(lw, log(AGE)),
 data=baltimore)
lm.morantest(obj1, lw)
lm.LMtests(obj1, lw, test="all")
obj2 <- lagmess(log(PRICE) ~ PATIO + log(AGE) + log(SQFT) +
 lag(lw, log(AGE)), data=baltimore, listw=lw)
summary(obj2)
obj3 <- lagsarlm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT) +
 lag(lw, log(AGE)), data=baltimore, listw=lw)
summary(obj3)
data(boston)
lw <- nb2listw(boston.soi)
gp2 <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2)
 +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, lw, method="Matrix")
summary(gp2)
gp2a <- lagmess(CMEDV ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2)
 +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, lw)
summary(gp2a)
```

---

lagsarlm *Spatial simultaneous autoregressive lag model estimation*

Description

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

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

where ρ is found by `optimize()` first, and β and other parameters by generalized least squares subsequently (one-dimensional search using optim performs badly on some platforms). In the spatial Durbin (mixed) model, the spatially lagged independent variables are added to X. Note that interpretation of the fitted coefficients should use impact measures, because of the feedback loops induced by the data generation process for this model. With one of the sparse matrix methods, larger numbers of observations can be handled, but the `interval=` argument may need be set when the weights are not row-standardised.

Usage

```
lagsarlm(formula, data = list(), listw,
na.action, type="lag", method="eigen", quiet=NULL,
zero.policy=NULL, interval=NULL, tol.solve=1.0e-10, trs=NULL,
control=list())
```

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 `options("na.action")`), 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 "spam" or "Matrix_J" for strictly symmetric weights lists of styles "B" and "C", or made symmetric by similarity (Ord, 1975, Appendix C) if possible for styles "W" and "S", using code from the spam or Matrix packages to calculate the determinant; “Matrix” and “spam_update” provide updating Cholesky decomposition methods; "LU" provides an alternative sparse matrix decomposition approach. In addition, there are "Chebyshev" and Monte Carlo "MC" approximate log-determinant methods; the Smirnov/Anselin (2009) trace approximation is available as "moments".
`quiet`	default NULL, use !verbose global option value; if FALSE, reports function values during optimization.
`zero.policy`	default NULL, use global option value; 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`	default is NULL, search interval for autoregressive parameter

`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

`trs` default NULL, if given, a vector of powered spatial weights matrix traces output by `trW`; when given, insert the asymptotic analytical values into the numerical Hessian instead of the approximated values; may be used to get around some problems raised when the numerical Hessian is poorly conditioned, generating NaNs in subsequent operations; the use of trs is recommended

`control` list of extra control arguments - see section below

Details

The asymptotic standard error of ρ 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="spam" or "Matrix". 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 ρ 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.

Versions of the package from 0.4-38 include numerical Hessian values where asymptotic standard errors are not available. This change has been introduced to permit the simulation of distributions for impact measures. The warnings made above with regard to variable scaling also apply in this case.

Note that the fitted() function for the output object assumes that 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 (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data.

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, otherwise approximate numeriacal Hessian-based values

`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
`opt`	object returned from numerical optimisation
`fitted.values`	Difference between residuals and response variable
`se.fit`	Not used yet
`formula`	model formula
`ase`	TRUE if method=eigen
`rho.se`	if ase=TRUE, the asymptotic standard error of ρ, otherwise approximate numeriacal Hessian-based value
`LMtest`	if ase=TRUE, the Lagrange Multiplier test for the absence of spatial autocorrelation in the lag model residuals
`resvar`	the asymptotic coefficient covariance matrix for (s2, rho, B)
`zero.policy`	zero.policy for this model
`aliased`	the aliased explanatory variables (if any)
`listw_style`	the style of the spatial weights used
`interval`	the line search interval used to find ρ
`fdHess`	the numerical Hessian-based coefficient covariance matrix for (rho, B) if computed
`optimHess`	if TRUE and fdHess returned, `optim` used to calculate Hessian at optimum
`insert`	if TRUE and fdHess returned, the asymptotic analytical values are inserted into the numerical Hessian instead of the approximated values, and its size increased to include the first row/column for sigma2
`LLNullLlm`	Log-likelihood of the null linear model
`timings`	processing timings
`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 ρ.

Control arguments

tol.opt: the desired accuracy of the optimization - passed to `optimize()` (default=square root of double precision machine tolerance, a larger root may be used needed, see help(boston) for an example)

fdHess: default NULL, then set to (method != "eigen") internally; use `fdHess` to compute an approximate Hessian using finite differences when using sparse matrix methods; used to make a coefficient covariance matrix when the number of observations is large; may be turned off to save resources if need be

optimHess: default FALSE, use `fdHess` from **nlme**, if TRUE, use `optim` to calculate Hessian at optimum

compiled_sse: default FALSE; logical value used in the log likelihood function to choose compiled code for computing SSE

Imult: default 2; used for preparing the Cholesky decompositions for updating in the Jacobian function

super: if NULL (default), set to FALSE to use a simplicial decomposition for the sparse Cholesky decomposition and method "Matrix_J", set to `as.logical(NA)` for method "Matrix", if TRUE, use a supernodal decomposition

cheb_q: default 5; highest power of the approximating polynomial for the Chebyshev approximation

MC_p: default 16; number of random variates

MC_m: default 30; number of products of random variates matrix and spatial weights matrix

spamPivot: default "MMD", alternative "RCM"

in_coef default 0.1, coefficient value for initial Cholesky decomposition in "spam_update"

type default "MC", used with method "moments"; alternatives "mult" and "moments", for use if `trs` is missing, `trW`

correct default TRUE, used with method "moments" to compute the Smirnov/Anselin correction term

trunc default TRUE, used with method "moments" to truncate the Smirnov/Anselin correction term

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; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York; LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton.

See Also

`lm`, `errorsarlm`, `eigenw`, `predict.sarlm`, `impacts.sarlm`, `residuals.sarlm`, `do_ldet`

Examples

```
data(oldcol)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), method="eigen", quiet=FALSE)
summary(COL.lag.eig, correlation=TRUE)
COL.lag.eig$fdHess
COL.lag.eig$resvar
W <- as(as_dgRMatrix_listw(nb2listw(COL.nb)), "CsparseMatrix")
trMatc <- trW(W, type="mult")
COL.lag.eig1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), control=list(fdHess=TRUE), trs=trMatc)
COL.lag.eig1$fdHess
system.time(COL.lag.M <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb), method="Matrix", quiet=FALSE))
summary(COL.lag.M)
impacts(COL.lag.M, listw=nb2listw(COL.nb))
## Not run:
system.time(COL.lag.sp <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb), method="spam", quiet=FALSE))
summary(COL.lag.sp)

## End(Not run)
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$CRIME[20:25] <- NA
COL.lag.NA <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 nb2listw(COL.nb), na.action=na.exclude,
 control=list(tol.opt=.Machine$double.eps^0.4))
COL.lag.NA$na.action
COL.lag.NA
resid(COL.lag.NA)
data(boston)
gp2mM <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed", method="Matrix")
summary(gp2mM)
W <- as(as_dgRMatrix_listw(nb2listw(boston.soi)), "CsparseMatrix")
trMatb <- trW(W, type="mult")
gp2mMi <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed", method="Matrix",
trs=trMatb)
summary(gp2mMi)
```

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. The as.spam.listw method converts a "listw" object to a sparse matrix as defined in the **spam** package, using listw2sn().

Usage

```
listw2sn(listw)
sn2listw(sn)
as.spam.listw(listw)
```

Arguments

listw	a listw object from for example nb2listw
sn	a spatial.neighbour object

Value

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

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

nb2listw

Examples

```
example(columbus)
col.listw <- nb2listw(col.gal.nb)
col.listw$neighbours[[1]]
col.listw$weights[[1]]
col.sn <- listw2sn(col.listw)
str(col.sn)
## Not run:
col.sp <- as.spam.listw(col.listw)
```

```
str(col.sp)

## End(Not run)
```

`lm.LMtests` *Lagrange Multiplier diagnostics for spatial dependence in linear models*

Description

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

Usage

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

Arguments

`model`	an object of class `lm` returned by `lm`, or optionally a vector of externally calculated residuals (run though `na.omit` if any NAs present) for use when only "LMerr" is chosen; weights and offsets should not be used in the `lm` object
`listw`	a `listw` object created for example by `nb2listw`, expected to be row-standardised (W-style)
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`test`	a character vector of tests requested chosen from 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()`
`naSubset`	default TRUE to subset listw object for omitted observations in model object (this is a change from earlier behaviour, when the `model$na.action` component was ignored, and the listw object had to be subsetted by hand)
`x`	object to be printed
`...`	printing arguments to be passed through

Details

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

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

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

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.LMtests(oldcrime.lm, nb2listw(COL.nb))
lm.LMtests(residuals(oldcrime.lm), nb2listw(COL.nb))
```

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

Description

Moran's I test for spatial autocorrelation in residuals from an estimated linear model (`lm()`). The helper function `listw2U()` constructs a weights list object corresponding to the sparse matrix $\frac{1}{2}(\mathbf{W} + \mathbf{W}'$

Usage

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

Arguments

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

Value

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

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

Note

The examples also show how to use permutation bootstrap if desired.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

lm.LMtests, lm

Examples

```
data(oldcol)
oldcrime1.lm <- lm(CRIME ~ 1, data = COL.OLD)
oldcrime.lm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD)
lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="W"))
lm.LMtests(oldcrime.lm, nb2listw(COL.nb, style="W"))
lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="S"))
lm.morantest(oldcrime1.lm, nb2listw(COL.nb, style="W"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
oldcrime.wlm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD,
 weights = I(1/AREA))
lm.morantest(oldcrime.wlm, nb2listw(COL.nb, style="W"),
 resfun=weighted.residuals)
lm.morantest(oldcrime.wlm, nb2listw(COL.nb, style="W"),
 resfun=rstudent)
if (require(boot)) {
  oldcrime.lmx <- lm(CRIME ~ HOVAL + INC, data = COL.OLD, x=TRUE)
  listw <- nb2listw(COL.nb, style="W")
  MoraneI.boot <- function(var, i, ...) {
    var <- var[i]
    lmres <- lm(var ~ oldcrime.lmx$x - 1)
    return(moran(x=residuals(lmres), ...)$I)
  }
  boot1 <- boot(residuals(oldcrime.lmx), statistic=MoraneI.boot, R=499,
    sim="permutation", listw=listw, n=length(listw$neighbours),
    S0=Szero(listw))
  zi <- (boot1$t0 - mean(boot1$t))/sqrt(var(boot1$t))
  boot1
  plot(boot1)
  cat("Bootstrap permutation standard deviate:", zi, "\n\n")
  lm.morantest(oldcrime.lm, nb2listw(COL.nb, style="W"))
}
```

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

Description

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

Usage

```
lm.morantest.exact(model, listw, zero.policy = NULL, alternative = "greater", spChk
## S3 method for class 'moranex'
print(x, ...)
```

Arguments

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

Value

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

`statistic`	the value of the 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.
`method`	a character string giving the method used.
`alternative`	a character string describing the alternative hypothesis.
`gamma`	eigenvalues (excluding zero values)
`oType`	usually set to "E"
`data.name`	a character string giving the name(s) of the data.
`df`	degrees of freedom

Author(s)

Markus Reder and Roger Bivand

See Also

`lm.morantest.sad`

Examples

```
eire <- readShapePoly(system.file("etc/shapes/eire.shp", package="spdep")[1],
  ID="names", proj4string=CRS("+proj=utm +zone=30 +units=km"))
eire.nb <- poly2nb(eire)
#data(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
lm.morantest.exact(e.lm, nb2listw(eire.nb))
lm.morantest.exact(e.lm, nb2listw(eire.nb), useTP=TRUE)
```

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

Description

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

Usage

```
lm.morantest.sad(model, listw, zero.policy=NULL, alternative="greater",
  spChk=NULL, resfun=weighted.residuals, tol=.Machine$double.eps^0.5,
  maxiter=1000, tol.bounds=0.0001, zero.tol = 1e-07, Omega=NULL,
  save.M=NULL, save.U=NULL)
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`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`alternative`	a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`resfun`	default: weighted.residuals; the function to be used to extract residuals from the `lm` object, may be `residuals`, `weighted.residuals`, `rstandard`, or `rstudent`
`tol`	the desired accuracy (convergence tolerance) for `uniroot`
`maxiter`	the maximum number of iterations for `uniroot`
`tol.bounds`	offset from bounds for `uniroot`
`zero.tol`	tolerance used to find eigenvalues close to absolute zero
`Omega`	A SAR process matrix may be passed in to test an alternative hypothesis, for example `Omega <- invIrW(listw, rho=0.1); Omega <- tcrossprod(Omega)`, `chol()` is taken internally
`save.M`	return the full M matrix for use in `spdep:::exactMoranAlt`
`save.U`	return the full U matrix for use in `spdep:::exactMoranAlt`
`x`	object to be printed
`object`	object to be summarised
`...`	arguments to be passed through

Details

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

Value

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

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

Author(s)

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

References

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

See Also

`lm.morantest`

Examples

```
eire <- readShapePoly(system.file("etc/shapes/eire.shp", package="spdep")[1],
  ID="names", proj4string=CRS("+proj=utm +zone=30 +units=km"))
eire.nb <- poly2nb(eire)
#data(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
summary(lm.morantest.sad(e.lm, nb2listw(eire.nb)))
e.wlm <- lm(OWNCONS ~ ROADACC, data=eire, weights=RETSALE)
lm.morantest(e.wlm, nb2listw(eire.nb), resfun=rstudent)
lm.morantest.sad(e.wlm, nb2listw(eire.nb), resfun=rstudent)
```

localG *G and Gstar local spatial statistics*

Description

The local spatial statistic G is calculated for each zone based on the spatial weights object used. The value returned is a Z-value, and may be used as a diagnostic tool. High positive values indicate the 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=NULL, 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`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`

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

`localmoran` *Local Moran's I statistic*

Description

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

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

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

Usage

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

Arguments

`x` a numeric vector the same length as the neighbours list in listw

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

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

`na.action` a function (default `na.fail`), can also be `na.omit` or `na.exclude` - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted. If `na.pass` is used, zero is substituted for NA values in calculating the spatial lag. (Note that na.exclude will only work properly starting from R 1.9.0, na.omit and na.exclude assign the wrong classes in 1.8.*)

`alternative` a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.

`p.adjust.method` a character string specifying the probability value adjustment for multiple tests, default "none"; see `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.

`mlvar` default TRUE: values of local Moran's I are reported using the variance of the variable of interest (sum of squared deviances over n), but can be reported as the sample variance, dividing by (n-1) instead; both are used in other implementations.

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

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

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

Description

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

Usage

```
localmoran.exact(model, select, nb, glist = NULL, style = "W", zero.policy = NULL,
localmoran.exact.alt(model, select, nb, glist = NULL, style = "W", zero.policy = NU
## S3 method for class 'localmoranex'
print(x, ...)
## S3 method for class 'localmoranex'
as.data.frame(x, row.names=NULL, optional=FALSE, ...)
```

Arguments

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

`useTP`	default FALSE, if TRUE, use truncation point in integration rather than upper=Inf, see Tiefelsdorf (2000), eq. 6.7, p.69
`truncErr`	when useTP=TRUE, pass truncation error to truncation point function
`zeroTreat`	when useTP=TRUE, pass zero adjustment to truncation point function
`x`	object to be printed
`row.names`	ignored argument to `as.data.frame.localmoranex`; row names assigned from localmoranex object
`optional`	ignored argument to `as.data.frame.localmoranex`; row names assigned from localmoranex object
`...`	arguments to be passed through

Value

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

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

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

Author(s)

Markus Reder and Roger Bivand

See Also

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

Examples

```
eire <- readShapePoly(system.file("etc/shapes/eire.shp", package="spdep")[1],
  ID="names", proj4string=CRS("+proj=utm +zone=30 +units=km"))
eire.nb <- poly2nb(eire)
#data(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
localmoran.sad(e.lm, nb=eire.nb)
localmoran.exact(e.lm, nb=eire.nb)
localmoran.exact(e.lm, nb=eire.nb, useTP=TRUE)
e.errorsar <- errorsarlm(OWNCONS ~ ROADACC, data=eire,
 listw=nb2listw(eire.nb))
localmoran.exact.alt(e.errorsar$lm.target, nb=eire.nb)
Omega <- invIrW(nb2listw(eire.nb), rho=0.6)
Omega1 <- tcrossprod(Omega)
localmoran.exact.alt(e.errorsar$lm.target, nb=eire.nb, Omega=Omega1)
localmoran.exact.alt(e.errorsar$lm.target, nb=eire.nb, Omega=Omega1, useTP=TRUE)
```

---

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

Description

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

Usage

```
localmoran.sad(model, select, nb, glist=NULL, style="W",
 zero.policy=NULL, alternative="greater", spChk=NULL,
 resfun=weighted.residuals, save.Vi=FALSE,
 tol = .Machine$double.eps^0.5, maxiter = 1000, tol.bounds=0.0001,
 save.M=FALSE, Omega = NULL)

print.localmoransad(x, ...)
summary.localmoransad(object, ...)
print.summary.localmoransad(x, ...)
listw2star(listw, ireg, style, n, D, a, zero.policy=NULL)
```

Arguments

`model` an object of class `lm` returned by `lm` (assuming no global spatial autocorrelation), or an object of class `sarlm` returned by a spatial simultaneous autoregressive model fit (assuming global spatial autocorrelation represented by the model

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

Details

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

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

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

Value

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

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

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

Examples

```
eire <- readShapePoly(system.file("etc/shapes/eire.shp", package="spdep")[1],
  ID="names", proj4string=CRS("+proj=utm +zone=30 +units=km"))
eire.nb <- poly2nb(eire)
#data(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
e.locmor <- summary(localmoran.sad(e.lm, nb=eire.nb))
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, weights=RETSALE)
```

```
e.locmorw1 <- summary(localmoran.sad(e.wlm, nb=eire.nb, resfun=weighted.residuals))
e.locmorw1
e.locmorw2 <- summary(localmoran.sad(e.wlm, nb=eire.nb, resfun=rstudent))
e.locmorw2
e.errorsar <- errorsarlm(OWNCONS ~ ROADACC, data=eire,
  listw=nb2listw(eire.nb))
e.errorsar
e.clocmor <- summary(localmoran.sad(e.errorsar$lm.target, nb=eire.nb))
e.clocmor
hist(e.clocmor[,"Pr. (Sad)"])
```

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. The spatial Hausman test is available for models fitted with `errorsarlm` and `GMerrorsar`.

Usage

```
LR.sarlm(x, y)
logLik.sarlm(object, ...)
LR1.sarlm(object)
Wald1.sarlm(object)
## S3 method for class 'sarlm'
Hausman.test(object, ..., tol=NULL)
## S3 method for class 'gmsar'
Hausman.test(object, ..., tol=NULL)
```

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()`
`...`	further arguments passed to or from other methods
`tol`	`tol` argument passed to `solve`, default NULL

Value

The tests return objects of class `htest` with:

`statistic`	value of statistic
`parameter`	degrees of freedom

`p.value`	Probability value
`estimate`	varies with test
`method`	description of test method

`logLik.sarlm()` returns an object of class `logLik` `LR1.sarlm`, `Hausman.sarlm` and `Wald1.sarlm` returm 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>`

References

LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton, pp. 61–63; Pace RK and LeSage J (2008) A spatial Hausman test. *Economics Letters* 101, 282–284.

See Also

`logLik.lm`, `anova.sarlm`

Examples

```
example(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)
Hausman.test(error)
```

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

Description

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

Usage

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

Arguments

x	A square non-negative matrix with no NAs representing spatial weights; may be a matrix of class "sparseMatrix"
row.names	row names to use for region IDs
style	default "M", unknown style; if not "M", passed to nb2listw to re-build the object

Value

A listw object with the following members:

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

nb2listw, nb2mat

Examples

```
example(columbus)
coords <- coordinates(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)
col005.w.mat.3T <- kronecker(diag(3), col005.w.mat)
col005.w.b.3T <- mat2listw(col005.w.mat.3T, style="W")
summary(col005.w.b.3T$neighbours)
W <- as(as_dgRMatrix_listw(nb2listw(col005, style="W", zero.policy=TRUE)), "CsparseMatrix")
col005.spM <- mat2listw(W)
summary(col005.spM$neighbours)
diffnb(col005, col005.spM$neighbours)
IW <- kronecker(Diagonal(3), W)
col005.spM.3T <- mat2listw(IW, style="W")
summary(col005.spM.3T$neighbours)
```

ME *Moran eigenvector GLM filtering*

Description

The Moran eigenvector filtering function is intended to remove spatial autocorrelation from the residuals of generalised linear models. It uses brute force eigenvector selection to reach a subset of such vectors to be added to the RHS of the GLM model to reduce residual autocorrelation to below the specified alpha value.

Usage

```
ME(formula, data, family = gaussian, weights, offset, listw,
 alpha=0.05, nsim=99, verbose=NULL, stdev=FALSE)
```

Arguments

`formula`	a symbolic description of the model to be fit
`data`	an optional data frame containing the variables in the model
`family`	a description of the error distribution and link function to be used in the model
`weights`	an optional vector of weights to be used in the fitting process
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting
`listw`	a `listw` object created for example by `nb2listw`
`alpha`	used as a stopping rule to choose all eigenvectors up to and including the one with a p-value exceeding alpha
`nsim`	number of permutations for permutation bootstrap for finding p-values
`verbose`	default NULL, use global option value; if TRUE report eigenvectors selected
`stdev`	if TRUE, p-value calculated from bootstrap permutation standard deviate using `pnorm` with alternative="greater", if FALSE the Hope-type p-value

Details

The eigenvectors for inclusion are chosen by calculating the empirical Moran's I values for the initial model plus each of the doubly centred symmetric spatial weights matrix eigenvectors in turn. Then the first eigenvector is chosen as that with the lowest Moran's I value. The procedure is repeated until the lowest remaining Moran's I value has a permutation-based probability value above alpha. The probability value is either Hope-type or based on using the mean and standard deviation of the permutations to calculate ZI based on the stdev argument.

Value

An object of class `ME_res`:

`selection` a matrix summarising the selection of eigenvectors for inclusion, with columns:

- **Eigenvector** number of selected eigenvector
- **ZI** permutation-based standardized deviate of Moran's I if stdev=TRUE
- **pr(ZI)** probability value: if stdev=TRUE of the permutation-based standardized deviate, if FALSE the Hope-type probability value, in both cases on-sided

The first row is the value at the start of the search

`vectors` a matrix of the selected eigenvectors in order of selection

Author(s)

Roger Bivand and Pedro Peres-Neto

References

Dray S, Legendre P and Peres-Neto PR (2005) Spatial modeling: a comprehensive framework for principle coordinate analysis of neigbbor matrices (PCNM), Ecological Modelling; Griffith DA and Peres-Neto PR (2006) Spatial modeling in ecology: the flexibility of eigenfunction spatial analyses.

See Also

`SpatialFiltering`, `glm`

Examples

```
## Not run:
example(columbus)
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)
lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL, data=columbus,
 nb=col.gal.nb, style="W", alpha=0.1, verbose=TRUE)
lagcol
lmlag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
anova(lmlag)
anova(lmbase, lmlag)
set.seed(123)
lagcol1 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
 listw=nb2listw(col.gal.nb), alpha=0.1, verbose=TRUE)
lagcol1
lmlag1 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol1), data=columbus)
anova(lmlag1)
anova(lmbase, lmlag1)
set.seed(123)
lagcol2 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
 listw=nb2listw(col.gal.nb), alpha=0.1, stdev=TRUE, verbose=TRUE)
lagcol2
lmlag2 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol2), data=columbus)
anova(lmlag2)
```

```
anova(lmbase, lmlag2)
example(nc.sids)
glmbase <- glm(SID74 ~ 1, data=nc.sids, offset=log(BIR74),
 family="poisson")
set.seed(123)
MEpois1 <- ME(SID74 ~ 1, data=nc.sids, offset=log(BIR74),
 family="poisson", listw=nb2listw(ncCR85_nb), alpha=0.2, verbose=TRUE)
MEpois1
glmME <- glm(SID74 ~ 1 + fitted(MEpois1), data=nc.sids, offset=log(BIR74),
 family="poisson")
anova(glmME, test="Chisq")
anova(glmbase, glmME, test="Chisq")
data(hopkins)
hopkins_part <- hopkins[21:36,36:21]
hopkins_part[which(hopkins_part > 0, arr.ind=TRUE)] <- 1
hopkins.rook.nb <- cell2nb(16, 16, type="rook")
glmbase <- glm(c(hopkins_part) ~ 1, family="binomial")
set.seed(123)
MEbinom1 <- ME(c(hopkins_part) ~ 1, family="binomial",
 listw=nb2listw(hopkins.rook.nb), alpha=0.2, verbose=TRUE)
glmME <- glm(c(hopkins_part) ~ 1 + fitted(MEbinom1), family="binomial")
anova(glmME, test="Chisq")
anova(glmbase, glmME, test="Chisq")

## End(Not run)
```

`moran` *Compute Moran's I*

Description

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

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

Usage

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

Arguments

`x`	a numeric vector the same length as the neighbours list in listw
`listw`	a `listw` object created for example by `nb2listw`
`n`	number of zones
`S0`	global sum of weights
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA

NAOK — if 'TRUE' then any 'NA' or 'NaN' or 'Inf' values in x are passed on to the foreign function. If 'FALSE', the presence of 'NA' or 'NaN' or 'Inf' values is regarded as an error.

Value

a list of

I — Moran's I

K — sample kurtosis of x

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

moran.test, moran.mc

Examples

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

---

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

Description

A permutation test for Moran's I statistic calculated by using nsim random permutations of x for the given spatial weighting scheme, to establish the rank of the observed statistic in relation to the nsim simulated values. 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=NULL, alternative="greater",
 na.action=na.fail, spChk=NULL, return_boot=FALSE)
```

Arguments

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

Value

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

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

Author(s)

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

References

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

See Also

`moran`, `moran.test`

Examples

```
data(oldcol)
colw <- nb2listw(COL.nb, style="W")
nsim <- 99
set.seed(1234)
sim1 <- moran.mc(COL.OLD$CRIME, listw=colw, nsim=nsim)
sim1
mean(sim1$res[1:nsim])
var(sim1$res[1:nsim])
summary(sim1$res[1:nsim])
MoranI.boot <- function(var, i, ...) {
var <- var[i]
return(moran(x=var, ...)$I)
}
set.seed(1234)
library(boot)
boot1 <- boot(COL.OLD$CRIME, statistic=MoranI.boot, R=nsim,
 sim="permutation", listw=colw, n=nrow(COL.OLD), S0=Szero(colw))
boot1
plot(boot1)
mean(boot1$t)
var(boot1$t)
summary(boot1$t)
colold.lags <- nblag(COL.nb, 3)
set.seed(1234)
sim2 <- moran.mc(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"), nsim=nsim)
summary(sim2$res[1:nsim])
sim3 <- moran.mc(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"), nsim=nsim)
summary(sim3$res[1:nsim])
```

---

moran.plot *Moran scatterplot*

Description

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

Usage

```
moran.plot(x, listw, zero.policy=NULL, spChk=NULL, labels=NULL,
 xlab=NULL, ylab=NULL, quiet=NULL, ...)
```

Arguments

- `x` — a numeric vector the same length as the neighbours list in listw
- `listw` — a `listw` object created for example by `nb2listw`
- `zero.policy` — default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
- `spChk` — should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
- `labels` — character labels for points with high influence measures, if set to FALSE, no labels are plotted for points with large influence
- `xlab` — label for x axis
- `ylab` — label for x axis
- `quiet` — default NULL, use !verbose global option value; if TRUE, output of summary of influence object suppressed
- `...` — 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(afcon$totcon, nb2listw(paper.nb),
 labels=as.character(afcon$name), pch=19)
moran.plot(scale(afcon$totcon), nb2listw(paper.nb),
 labels=as.character(afcon$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=NULL,
 alternative="greater", rank = FALSE, na.action=na.fail, spChk=NULL, adjust.n=TRUE)
```

Arguments

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

Value

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

`statistic`	the value of the standard deviate of Moran's I.
`p.value`	the p-value of the test.

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

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

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

Examples

```
data(oldcol)
coords.OLD <- cbind(COL.OLD$X, COL.OLD$Y)
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="B"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="C"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="S"))
moran.test(COL.OLD$CRIME, nb2listw(COL.nb, style="W"),
 randomisation=FALSE)
colold.lags <- nblag(COL.nb, 3)
moran.test(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
 style="W"))
moran.test(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
 style="W"))
print(is.symmetric.nb(COL.nb))
COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
print(is.symmetric.nb(COL.k4.nb))
moran.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"))
moran.test(COL.OLD$CRIME, nb2listw(COL.k4.nb, style="W"),
 randomisation=FALSE)
cat("Note: non-symmetric weights matrix, use listw2U()")
```

```
moran.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")))
moran.test(COL.OLD$CRIME, listw2U(nb2listw(COL.k4.nb,
 style="W")), randomisation=FALSE)
ranks <- rank(COL.OLD$CRIME)
names(ranks) <- rownames(COL.OLD)
moran.test(ranks, nb2listw(COL.nb, style="W"), rank=TRUE)
crime <- COL.OLD$CRIME
is.na(crime) <- sample(1:length(crime), 10)
res <- try(moran.test(crime, nb2listw(COL.nb, style="W"),
 na.action=na.fail))
res
moran.test(crime, nb2listw(COL.nb, style="W"), zero.policy=TRUE,
 na.action=na.omit)
moran.test(crime, nb2listw(COL.nb, style="W"), zero.policy=TRUE,
 na.action=na.exclude)
moran.test(crime, nb2listw(COL.nb, style="W"), na.action=na.pass)
```

`mstree` *Find the minimal spanning tree*

Description

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

Usage

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

Arguments

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

Details

The minimum spanning tree algorithm.

Input a connected graph.

Begin a empty set of nodes.

Add an arbitrary note in this set.

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

The set of edges is a minimum spanning tree.

Value

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

Author(s)

Renato M. Assuncao and Elias T. Krainski

References

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

Examples

```
### loading data
bh <- readShapePoly(system.file("etc/shapes/bhicv.shp",
      package="spdep")[1])
### data padronized
dpad <- data.frame(scale(bh@data[,5:8]))

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

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

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

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

dim(mst.bh)

head(mst.bh)
tail(mst.bh)

### the mstree plot
par(mar=c(0,0,0,0))
plot(mst.bh, coordinates(bh), col=2,
     cex.lab=.7, cex.circles=0.035, fg="blue")
plot(bh, border=gray(.5), add=TRUE)
```

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

Description

Set operations on neighbors list objects

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

Nicholas Lewin-Koh `<nikko@hailmail.net>`

See Also

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

Examples

```
example(columbus)
coords <- coordinates(columbus)
col.tri.nb <- tri2nb(coords)
oldpar <- par(mfrow=c(1,2))
col.soi.nb <- graph2nb(soi.graph(col.tri.nb, coords))
plot(columbus, border="grey")
plot(col.soi.nb, coords, add=TRUE)
title(main="Sphere of Influence Graph")
plot(columbus, border="grey")
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)
```

`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.character(ID)))` must be identical to `sort(as.character(attr(nb, "region.id")));` same length as `row.names` if provided.
`row.names`	character vector of observation ids to be added to the neighbours list as attribute `region.id`, default `seq(1, nrow(x))`; same length as `ID` if provided.

Details

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

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knn2nb, dnearneigh, cell2nb, tri2nb, poly2nb

Examples

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

## End(Not run)
```

---

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, proj4string=CRS(as.character(NA)))
listw2lines(listw, coords, proj4string=CRS(as.character(NA)))
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

`proj4string`	Object of class CRS; holding a valid proj4 string
`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 `writeLinesShape` 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 data slot of the SpatialLinesDataFrame object into a spatial.neighbour object, which is then converted into a weights list object.

Value

`nb2lines` and `listw2lines` return a SpatialLinesDataFrame object; its data slot contains 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`, `readShapeLines`

Examples

```
example(columbus)
coords <- coordinates(columbus)
res <- listw2lines(nb2listw(col.gal.nb), coords)
summary(res)
fn <- paste(tempdir(), "nbshape", sep="/")
writeLinesShape(res, fn=fn)
inMap <- readShapeLines(fn)
summary(inMap)
diffnb(sn2listw(df2sn(as(inMap, "data.frame")))$neighbours, col.gal.nb)
```

`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=NULL)
```

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`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors

Details

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

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

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

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

```
example(columbus)
coords <- coordinates(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$weights))
# introducing other conditions into weights - only earlier sales count
# see http://sal.uiuc.edu/pipermail/openspace/2005-October/000610.html
data(baltimore)
set.seed(211)
dates <- sample(1:500, nrow(baltimore), replace=TRUE)
nb_15nn <- knn2nb(knearneigh(cbind(baltimore$X, baltimore$Y), k=15))
glist <- vector(mode="list", length=length(nb_15nn))
for (i in seq(along=nb_15nn))
  glist[[i]] <- ifelse(dates[i] > dates[nb_15nn[[i]]], 1, 0)
listw_15nn_dates <- nb2listw(nb_15nn, glist=glist, style="B")
which(lag(listw_15nn_dates, baltimore$PRICE) == 0.0)
which(sapply(glist, sum) == 0)
ex <- which(sapply(glist, sum) == 0)[1]
dates[ex]
dates[nb_15nn[[ex]]]
```

nb2mat *Spatial weights matrices for neighbours lists*

Description

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

Usage

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

Arguments

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

Details

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

The function leaves matrix rows as zero for any regions with zero neighbours fore zero.policy TRUE. These will in turn generate lag values of zero, equivalent to the sum of products of the zero row `t(rep(0, length=length(neighbours))) %*% x`, for 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

```
example(columbus)
coords <- coordinates(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)))
```

---

nb2WB *Output spatial weights for WinBUGS*

Description

Output spatial weights for WinBUGS

Usage

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

Arguments

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

Value

A list suitable for convering using dput for WinBUGS

Author(s)

Virgilio Gomez-Rubio

References

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

See Also

dput

Examples

```
example(columbus)
x <- nb2WB(col.gal.nb)
dput(x, control=NULL)
x <- listw2WB(nb2listw(col.gal.nb))
dput(x, control=NULL)
```

---

nbcosts *Compute cost of edges*

Description

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

Usage

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

Arguments

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

Value

A object of nbdist class. See nbdists for details.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as `nbdists`, `nb2listw`

`nbdists` *Spatial link distance measures*

Description

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

Usage

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

Arguments

`nb`	an object of class `nb`
`coords`	matrix of point coordinates or a SpatialPoints object
`longlat`	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if coords is a SpatialPoints object, the value is taken from the object itself

Value

A list with class `nbdist`

Author(s)

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

See Also

`summary.nb`, `nb2listw`

Examples

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

nblag *Higher order neighbours lists*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

See Also

`summary.nb`

Examples

```
example(columbus)
coords <- coordinates(columbus)
summary(col.gal.nb, coords)
plot(columbus, border="grey")
plot(col.gal.nb, coords, add=TRUE)
title(main="GAL order 1 (black) and 2 (red) links")
col.lags <- nblag(col.gal.nb, 2)
lapply(col.lags, print)
summary(col.lags[[2]], coords)
plot(col.lags[[2]], coords, add=TRUE, col="red", lty=2)
cuml <- nblag_cumul(col.lags)
cuml
```

nc.sids *North Carolina SIDS data*

Description

(Use `example(nc.sids)` to read the data set from shapefile, together with import of two different list of neighbours).

The `nc.sids` data frame has 100 rows and 21 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:

SP_ID SpatialPolygons ID

CNTY_ID county ID

east eastings, county seat, miles, local projection

north northings, county seat, miles, local projection

L_id Cressie and Read (1985) L index

M_id Cressie and Read (1985) M index

names County names

AREA County polygon areas in degree units

PERIMETER County polygon perimeters in degree units

CNTY_ Internal county ID

NAME County names

FIPS County ID

FIPSNO County ID

CRESS_ID Cressie papers 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

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

```
nc.sids <- readShapePoly(system.file("etc/shapes/sids.shp", package="spdep")[1],
  ID="FIPSNO", proj4string=CRS("+proj=longlat +ellps=clrk66"))
rn <- sapply(slot(nc.sids, "polygons"), function(x) slot(x, "ID"))
ncCC89_nb <- read.gal(system.file("etc/weights/ncCC89.gal", package="spdep")[1],
  region.id=rn)
ncCR85_nb <- read.gal(system.file("etc/weights/ncCR85.gal", package="spdep")[1],
  region.id=rn)
## Not run:
plot(nc.sids, border="grey")
plot(ncCR85_nb, coordinates(nc.sids), add=TRUE, col="blue")
plot(nc.sids, border="grey")
plot(ncCC89_nb, coordinates(nc.sids), add=TRUE, col="blue")

## End(Not run)
```

NY_data   *New York leukemia data*

Description

New York leukemia data taken from the data sets supporting Waller and Gotway 2004 (the data should be loaded by running `example(NY_data)` to demonstrate spatial data import techniques).

Usage

```
data(NY_data)
```

Format

A data frame with 281 observations on the following 12 variables, and the binary coded spatial weights used in the source.

`AREANAME` name of census tract

`AREAKEY` unique FIPS code for each tract

`X` x-coordinate of tract centroid (in km)

`Y` y-coordinate of tract centroid (in km)

`POP8` population size (1980 U.S. Census)

`TRACTCAS` number of cases 1978-1982

PROPCAS proportion of cases per tract

PCTOWNHOME percentage of people in each tract owning their own home

PCTAGE65P percentage of people in each tract aged 65 or more

Z ransformed propoprtions

AVGIDIST average distance between centroid and TCE sites

PEXPOSURE "exposure potential": inverse distance between each census tract centroid and the nearest TCE site, IDIST, transformed via log(100*IDIST)

Details

The examples section shows how the DBF files from the book website for Chapter 9 were converted into the nydata data frame and the listw_NY spatial weights list.

Source

http://www.sph.emory.edu/~lwaller/ch9index.htm

References

Waller, L. and C. Gotway (2004) *Applied Spatial Statistics for Public Health Data*. New York: John Wiley and Sons.

Examples

```
## NY leukemia
nydata <- read.dbf(system.file("etc/misc/nydata.dbf", package="spdep")[1])
coordinates(nydata) <- c("X", "Y")
nyadjmat <- as.matrix(read.dbf(system.file("etc/misc/nyadjwts.dbf",
 package="spdep")[1])[-1])
ID <- as.character(names(read.dbf(system.file("etc/misc/nyadjwts.dbf",
 package="spdep")[1]))[-1])
identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEY), 2, 10))
nyadjlw <- mat2listw(nyadjmat, as.character(nydata$AREAKEY))
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
```

---

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

Description

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

Usage

```
data(oldcol)
```

Format

This data frame contains the following columns:

AREA 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.

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

plot.mst — *Plot the Minimum Spanning Tree*

Description

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

Usage

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

Arguments

`x`	Object of `mst` class.
`coords`	A two column matrix with the coordinates of nodes.
`label.areas`	A vector with the labels of nodes
`cex.circles`	The length of circles to plot.
`cex.labels`	The length of nodes labels ploted.
`...`	Further arguments passed to plotting funcitons.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as `skater` and `mstree`

Examples

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

plot.nb — *Plot a neighbours list*

Description

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

Usage

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

Arguments

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

Author(s)

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

See Also

`summary.nb`

Examples

```
example(columbus)
coords <- coordinates(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)
```

`plot.skater` *Plot the object of skater class*

Description

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

Usage

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

Arguments

x	An object of skater class.
coords	A matrix of two colums with coordinates of nodes.
label.areas	A vector of labels of nodes.
groups.colors	A vector with colors of groups ou sub-graphs.
cex.circles	The length of circles with represent the nodes.
cex.labels	The length of labels of nodes.
...	Further arguments passed to plotting funcitons.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as skater and mstree

Examples

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

---

poly2nb *Construct neighbours list from polygon list*

Description

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

Usage

```
poly2nb(pl, row.names = NULL, snap=sqrt(.Machine$double.eps),
 queen=TRUE, useC=TRUE, foundInBox=NULL)
```

Arguments

- `pl` — list of polygons of class extending `SpatialPolygons`, or an object of a class `polylist`, which is retained for compatibility reasons
- `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; note that more than one shared boundary point does not necessarily mean a shared boundary line
- `useC` — default TRUE, doing the work loop in C, may be set to false to revert to R code calling two C functions in an `n*k` work loop, where `k` is the average number of candidate neighbours
- `foundInBox` — default NULL using R code, possibly parallelised if a **snow** cluster is available, otherwise a list of length `(n-1)` with integer vectors of candidate neighbours `(j > i)`, or NULL if all candidates were `(j < i)` (as created by the `poly_findInBoxGEOS` function in **rgeos** for clean polygons)

Value

A neighbours list with class `nb`

Note

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

Author(s)

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

See Also

`summary.nb`

Examples

```
example(columbus)
coords <- coordinates(columbus)
xx <- poly2nb(columbus)
dxx <- diffnb(xx, col.gal.nb)
plot(columbus, border="grey")
plot(col.gal.nb, coords, add=TRUE)
plot(dxx, coords, add=TRUE, col="red")
title(main=paste("Differences (red) in Columbus GAL weights (black)",
 "and polygon generated queen weights", sep="\n"))
```

```
xxx <- poly2nb(columbus, queen=FALSE)
dxxx <- diffnb(xxx, col.gal.nb)
plot(columbus, border = "grey")
plot(col.gal.nb, coords, add = TRUE)
plot(dxxx, coords, add = TRUE, col = "red")
title(main=paste("Differences (red) in Columbus GAL weights (black)",
 "and polygon generated rook weights", sep="\n"))
cards <- card(xx)
maxconts <- which(cards == max(cards))
if(length(maxconts) > 1) maxconts <- maxconts[1]
fg <- rep("grey", length(cards))
fg[maxconts] <- "red"
fg[xx[[maxconts]]] <- "green"
plot(columbus, col=fg)
title(main="Region with largest number of contiguities")
example(nc.sids)
system.time(xxnb <- poly2nb(nc.sids))
plot(nc.sids)
plot(xxnb, coordinates(nc.sids), add=TRUE, col="blue")
```

---

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 = NULL, ...)
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`	default NULL, use global option value; 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 (Cressie 1993, p. 564). 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 = ρWy.

This approach differs from the design choices made in other software, for example GeoDa, which does not use observations of the response variable, and corresponds to the newdata situation described below.

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:

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

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; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York.

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, alternative="less")
```

Arguments

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

Details

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

Value

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

Author(s)

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

References

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

See Also

`EBest`, `EBlocal`, `ppois`

Examples

```
example(auckland)
res <- probmap(auckland$M77_85, 9*auckland$Und5_81)
rt <- sum(auckland$M77_85)/sum(9*auckland$Und5_81)
ppois_pmap <- numeric(length(auckland$Und5_81))
for (i in seq(along=ppois_pmap)) {
ppois_pmap[i] <- poisson.test(auckland$M77_85[i], r=rt,
  T=(9*auckland$Und5_81[i]), alternative="less")$p.value
}
all.equal(ppois_pmap, res$pmap)
brks <- c(-Inf,2,2.5,3,3.5,Inf)
cols <- grey(6:2/7)
plot(auckland, col=cols[findInterval(res$raw*1000, brks, all.inside=TRUE)])
legend("bottomleft", 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(auckland, col=cols[findInterval(res$relRisk, brks, all.inside=TRUE)])
legend("bottomleft", fill=cols, legend=leglabs(brks), bty="n")
title(main="Standardised mortality ratios for Auckland child deaths")
brks <- c(0,0.05,0.1,0.2,0.8,0.9,0.95,1)
cols <- cm.colors(7)
plot(auckland, col=cols[findInterval(res$pmap, brks, all.inside=TRUE)])
legend("bottomleft", fill=cols, legend=leglabs(brks), bty="n")
title(main="Poisson probabilities for Auckland child mortality")
```

prunecost *Compute cost of prune each edge*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as `prunemst`

Examples

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

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

prunecost(e, d)
```

---

prunemst *Prune a Minimun Spanning Tree*

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as `mstree`

Examples

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

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

Description

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

Usage

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

Arguments

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

Details

Luc Anselin (2003): Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, 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). The body of the files after any headers should have three columns separated by white space, and the third column must be numeric in the locale of the reading platform (correct decimal separator).

Usage

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

Arguments

`file`	name of file with weights data
`region.id`	region IDs
`sn`	a `spatial.neighbour` object
`shpfile`	character string: if not given Shapefile name taken from GWT file for this dataset
`ind`	character string: region id indicator field name
`useInd`	default FALSE, if TRUE, write `region.id` attribute ID key tags to output file (use in OpenGeoDa will depend on the shapefile having the field named in the `ind` argument matching the exported tags)
`legacy`	default FALSE; if TRUE, header has single field with number of observations only

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"))
diffnb(gwt1, gwt2)
data(oldcol)
tmpMAT <- tempfile()
COL.W <- nb2listw(COL.nb)
write.sn2dat(listw2sn(COL.W), tmpMAT)
listwmat1 <- read.dat2listw(tmpMAT)
diffnb(listwmat1$neighbours, COL.nb, verbose=TRUE)
listwmat2 <- read.dat2listw(system.file("etc/weights/wmat.dat",
 package="spdep")[1])
diffnb(listwmat1$neighbours, listwmat2$neighbours, verbose=TRUE)
```

`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

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

F. Guillaume Blanchet

Examples

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

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

for(i in rad){
coords.rot<-Rotation(coords,i)
plot(coords.rot)
}

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

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

sacsarlm	*Spatial simultaneous autoregressive SAC model estimation*

Description

Maximum likelihood estimation of spatial simultaneous autoregressive "SAC/SARAR" models of the form:

$$y = \rho W1y + X\beta + u, u = \lambda W2u + \varepsilon$$

where ρ and λ are found by `nlminb` or `optim()` first, and β and other parameters by generalized least squares subsequently

Usage

```
sacsarlm(formula, data = list(), listw, listw2 = NULL, na.action, type="sac",
 method = "eigen", quiet = NULL, zero.policy = NULL, tol.solve = 1e-10,
 llprof=NULL, control = list())
```

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`
`listw2`	a `listw` object created for example by `nb2listw`, if not given, set to the same spatial weights as the `listw` argument
`na.action`	a function (default `options("na.action")`), 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 "sac", may be set to "sacmixed" for the Manski model to include the spatially lagged independent variables added to X using `listw`; when "sacmixed", 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 "spam" or "Matrix" for strictly symmetric weights lists of styles "B" and "C", or made symmetric by similarity (Ord, 1975, Appendix C) if possible for styles "W" and "S", using code from the spam or Matrix packages to calculate the determinant; "LU" provides an alternative sparse matrix decomposition approach. In addition, there are "Chebyshev" and Monte Carlo "MC" approximate log-determinant methods.

`quiet`	default NULL, use !verbose global option value; if FALSE, reports function values during optimization.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `sacsarlm()` to terminate with an error
`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
`llprof`	default NULL, can either be an integer, to divide the feasible ranges into a grid of points, or a two-column matrix of spatial coefficient values, at which to evaluate the likelihood function
`control`	list of extra control arguments - see section below

Details

Because numerical optimisation is used to find the values of lambda and rho, care needs to be shown. It has been found that the surface of the 2D likelihood function often forms a "banana trench" from (low rho, high lambda) through (high rho, high lambda) to (high rho, low lambda) values. In addition, sometimes the banana has optima towards both ends, one local, the other global, and conseqently the choice of the starting point for the final optimization becomes crucial. The default approach is not to use just (0, 0) as a starting point, nor the (rho, lambda) values from `gstsls`, which lie in a central part of the "trench", but either four values at (low rho, high lambda), (0, 0), (high rho, high lambda), and (high rho, low lambda), and to use the best of these start points for the final optimization. Optionally, nine points can be used spanning the whole (lower, upper) space.

Value

A list object of class `sarlm`

`type`	"sac"
`rho`	lag simultaneous autoregressive lag coefficient
`lambda`	error simultaneous autoregressive error coefficient
`coefficients`	GLS coefficient estimates
`rest.se`	asymptotic standard errors if ase=TRUE, otherwise approximate numeriacal Hessian-based values
`ase`	TRUE if method=eigen
`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
`opt`	object returned from numerical optimisation
`pars`	starting parameter values for final optimization, either given or found by trial point evaluation
`mxs`	if default input pars, optimal objective function values at trial points
`fitted.values`	Difference between residuals and response variable
`se.fit`	Not used yet
`formula`	model formula
`rho.se`	if ase=TRUE, the asymptotic standard error of ρ, otherwise approximate numeriacal Hessian-based value
`lambda.se`	if ase=TRUE, the asymptotic standard error of λ
`resvar`	the asymptotic coefficient covariance matrix for (s2, rho, lambda, B)
`zero.policy`	zero.policy for this model
`aliased`	the aliased explanatory variables (if any)
`LLNullLlm`	Log-likelihood of the null linear model
`fdHess`	the numerical Hessian-based coefficient covariance matrix for (rho, B) if computed
`resvar`	
`optimHess`	if TRUE and fdHess returned, `optim` used to calculate Hessian at optimum
`timings`	processing timings
`na.action`	(possibly) named vector of excluded or omitted observations if non-default na.action argument used

Control arguments

fdHess: default NULL, then set to (method != "eigen") internally; use `fdHess` to compute an approximate Hessian using finite differences when using sparse matrix methods; used to make a coefficient covariance matrix when the number of observations is large; may be turned off to save resources if need be

optimHess: default FALSE, use `fdHess` from **nlme**, if TRUE, use `optim` to calculate Hessian at optimum

LAPACK: default FALSE; logical value passed to `qr` in the SSE log likelihood function

Imult: default 2; used for preparing the Cholesky decompositions for updating in the Jacobian function

super: default FALSE using a simplicial decomposition for the sparse Cholesky decomposition, if TRUE, use a supernodal decomposition

cheb_q: default 5; highest power of the approximating polynomial for the Chebyshev approximation

MC_p: default 16; number of random variates

MC_m: default 30; number of products of random variates matrix and spatial weights matrix

opt_method: default "nlminb", may be set to "L-BFGS-B" to use box-constrained optimisation in `optim`

opt_control: default `list()`, a control list to pass to `nlminb` or `optim`

pars: default `NULL`, for which five trial starting values spanning the lower/upper range are tried and the best selected, starting values of ρ and λ

npars default integer `4L`, four trial points; if not default value, nine trial points

lower: default `c(-1.0, -1.0)`, lower bounds on ρ and λ

upper: default `c(1.0, 1.0)`, upper bounds on ρ and λ

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Anselin, L. 1988 *Spatial econometrics: methods and models.* (Dordrecht: Kluwer); LeSage J and RK Pace (2009) *Introduction to Spatial Econometrics*. CRC Press, Boca Raton

See Also

`help`,

Examples

```
data(oldcol)
COL.sacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"))
summary(COL.sacW.eig, correlation=TRUE)
W <- as(as_dgRMatrix_listw(nb2listw(COL.nb, style="W")), "CsparseMatrix")
trMatc <- trW(W, type="mult")
summary(impacts(COL.sacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"), type="sacmixed")
summary(COL.msacW.eig, correlation=TRUE)
summary(impacts(COL.msacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
```

set.spChkOption *Control checking of spatial object IDs*

Description

Provides support for checking the mutual integrity of spatial neighbour weights and spatial data; similar mechanisms are used for passing global verbose and zero.policy options, and for providing access to a running cluster for embarrassingly parallel tasks.

Usage

```
set.spChkOption(check)
get.spChkOption()
chkIDs(x, listw)
spNamedVec(var, data)
set.VerboseOption(check)
get.VerboseOption()
set.ZeroPolicyOption(check)
get.ZeroPolicyOption()
set.ClusterOption(cl)
get.ClusterOption()
set.rlecuyerSeedOption(seed)
get.rlecuyerSeedOption()
```

Arguments

`check`	a logical value, TRUE or FALSE
`cl`	a cluster object created by `makeCluster` in **snow**
`seed`	an integer vector of length 6 to seed the Lecuyer streams for cluster RNG
`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 Matrix 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

`lagsarlm`, `errorsarlm`

Examples

```
data(oldcol)
COL.W <- nb2listw(COL.nb, style="W")
COL.S <- nb2listw(COL.nb, style="S")
sum(log(1 - 0.5 * eigenw(COL.W)))
sum(log(1 - 0.5 * eigenw(similar.listw(COL.W))))
W_J <- as(as_dsTMatrix_listw(similar.listw(COL.W)), "CsparseMatrix")
I <- as_dsCMatrix_I(dim(W_J)[1])
c(determinant(I - 0.5 * W_J, logarithm=TRUE)$modulus)
sum(log(1 - 0.5 * eigenw(COL.S)))
sum(log(1 - 0.5 * eigenw(similar.listw(COL.S))))
W_J <- as(as_dsTMatrix_listw(similar.listw(COL.S)), "CsparseMatrix")
c(determinant(I - 0.5 * W_J, logarithm=TRUE)$modulus)
```

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

Description

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

Usage

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

Arguments

`edges`	A matrix with 2 colums with each row is an edge
`data`	A data.frame with data observed over nodes.
`ncuts`	The number of cuts
`crit`	A numeric or integer with criteria for groups. Example: minimum population size.
`vec.crit`	A vector for evaluating critera.
`method`	Character for declare the distance method. For "euclidean", "maximum", "manhattan", "canberra", "binary" and "minkowski", see `dist` for details, because this function as used to compute the distance. If `method="mahalanobis"`, the mahalanobis distance is computed between neighbour areas. If `method="other"`, any function must be informed in `otherfun` argument.
`p`	The power of the Minkowski distance.
`cov`	The covariance matrix used to compute the mahalanobis distance.
`inverted`	logical. If 'TRUE', 'cov' is supposed to contain the inverse of the covariance matrix.
`otherfun`	A user defined function to compute the distance

Details

to do

Value

A object of `skater` class with:

`groups`	A vector with length equal the number of nodes. Each position identifies the group of node

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

Author(s)

Renato M. Assuncao and Elias T. Krainski

References

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

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

See Also

See Also as mstree

Examples

```
### loading data
bh <- readShapePoly(system.file("etc/shapes/bhicv.shp",
      package="spdep")[1])
### data padronized
dpad <- data.frame(scale(bh@data[,5:8]))

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

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

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

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

### the mstree plot
par(mar=c(0,0,0,0))
plot(mst.bh, coordinates(bh), col=2,
     cex.lab=.7, cex.circles=0.035, fg="blue")
plot(bh, border=gray(.5), add=TRUE)

### three groups with no restriction
res1 <- skater(mst.bh[,1:2], dpad, 2)
```

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

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

### groups frequency
table(res1$groups)
table(res2$groups)
table(res3$groups)

### the skater plot
par(mar=c(0,0,0,0))
plot(res1, coordinates(bh), cex.circles=0.035, cex.lab=.7)

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

### length groups frequency
table(res1$groups)
table(res1b$groups)

### the skater plot, using other colors
plot(res1b, coordinates(bh), cex.circles=0.035, cex.lab=.7,
     groups.colors=colors()[(1:length(res1b$ed))*10])

### the Spatial Polygons plot
plot(bh, col=heat.colors(4)[res1b$groups])
```

`sp.correlogram` *Spatial correlogram*

Description

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

Usage

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

Arguments

`neighbours`	an object of class `nb`
`var`	a numeric vector

`order`	maximum lag order
`method`	"corr" for correlation, "I" for Moran's I, "C" for Geary's C
`style`	`style` can take values W, B, C, and S
`randomisation`	variance of I or C calculated under the assumption of randomisation, if FALSE normality
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
`spChk`	should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`
`x`	an object from `sp.correlogram()` of class `spcor`
`p.adj.method`	correction method as in `p.adjust`
`main`	an overall title for the plot
`ylab`	a title for the y axis
`ylim`	the y limits of the plot
`...`	further arguments passed through

Details

The print function also calculates the standard deviates of Moran's I or Geary's C and a two-sided probability value, optionally using `p.adjust` to correct by the nymber of lags. The plot function plots a bar from the estimated Moran's I, or Geary's C value to +/- twice the square root of its variance (in previous releases only once, not twice).

Value

returns a list of class `spcor`:

`res`	for "corr" a vector of values; for "I", a matrix of estimates of "I", expectations, and variances
`method`	"I" or "corr"
`cardnos`	list of tables of neighbour cardinalities for the lag orders used
`var`	variable name

Author(s)

Roger Bivand, `<Roger.Bivand@nhh.no>`

References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion, pp. 118–122, Martin, R. L., Oeppen, J. E. 1975 The identification of regional forecasting models using space-time correlation functions, *Transactions of the Institute of British Geographers*, 66, 95–118.

See Also

nblag, moran, p.adjust

Examples

```
example(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)
cspc <- sp.correlogram(ncCC89_nb, tr.SIDS74, order=8, method="corr",
 zero.policy=TRUE)
print(cspc)
plot(cspc)
Ispc <- sp.correlogram(ncCC89_nb, tr.SIDS74, order=8, method="I",
 zero.policy=TRUE)
print(Ispc)
print(Ispc, "bonferroni")
plot(Ispc)
Cspc <- sp.correlogram(ncCC89_nb, tr.SIDS74, order=8, method="C",
 zero.policy=TRUE)
print(Cspc)
print(Cspc, "bonferroni")
plot(Cspc)
drop.no.neighs <- !(1:length(ncCC89_nb) %in% which(card(ncCC89_nb) == 0))
sub.ncCC89.nb <- subset(ncCC89_nb, drop.no.neighs)
plot(sp.correlogram(sub.ncCC89.nb, subset(tr.SIDS74,  drop.no.neighs),
 order=8, method="corr"))
```

---

sp.mantel.mc *Mantel-Hubert spatial general cross product statistic*

Description

A permutation test for the spatial general cross product statistic with Moran ($C_{ij} = z_i z_j$), Geary ($C_{ij} = (z_i - z_j)^2$), and Sokal ($C_{ij} = |z_i - z_j|$) criteria, for $z_i = (x_i - \bar{x})/\sigma_x$. plot.mc.sim is a helper function to plot the outcomes of the permutation test.

Usage

```
sp.mantel.mc(var, listw, nsim, type = "moran", zero.policy = NULL,
 alternative = "greater", spChk=NULL, return_boot=FALSE)
plot.mc.sim(x, xlim, xlab, main, sub, ..., ptype="density")
```

Arguments

var	a numeric vector the same length as the neighbours list in listw
listw	a listw object created for example by nb2listw
nsim	number of permutations

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

Value

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

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

Author(s)

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

References

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

See Also

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

Examples

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

SpatialFiltering *Semi-parametric spatial filtering*

Description

The function selects eigenvectors in a semi-parametric spatial filtering approach to removing spatial dependence from linear models. Selection is by brute force by finding the single eigenvector reducing the standard variate of Moran's I for regression residuals most, and continuing until no candidate eigenvector reduces the value by more than `tol`. It returns a summary table from the selection process and a matrix of selected eigenvectors for the specified model.

Usage

```
SpatialFiltering(formula, lagformula, data = list(), nb, glist = NULL, style = "C",
 zero.policy = NULL, tol = 0.1, zerovalue = 1e-04, ExactEV = FALSE,
 symmetric = TRUE, alpha=NULL, alternative="two.sided", verbose=NULL)
```

Arguments

`formula`	a symbolic description of the model to be fit, assuming a spatial error representation; when lagformula is given, it should include only the response and the intercept term
`lagformula`	An extra one-sided formula to be used when a spatial lag representation is desired; the intercept is excluded within the function if present because it is part of the formula argument, but excluding it explicitly in the lagformula argument in the presence of factors generates a collinear model matrix
`data`	an optional data frame containing the variables in the model
`nb`	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`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
`tol`	tolerance value for convergence of spatial filtering

`zerovalue`	eigenvectors with eigenvalues of an absolute value smaller than zerovalue will be excluded in eigenvector search
`ExactEV`	Set ExactEV=TRUE to use exact expectations and variances rather than the expectation and variance of Moran's I from the previous iteration, default FALSE
`symmetric`	Should the spatial weights matrix be forced to symmetry, default TRUE
`alpha`	if not NULL, used instead of the tol= argument as a stopping rule to choose all eigenvectors up to and including the one with a probability value exceeding alpha.
`alternative`	a character string specifying the alternative hypothesis, must be one of greater, less or two.sided (default).
`verbose`	default NULL, use global option value; if TRUE report eigenvectors selected

Value

An `SFResult` object, with:

`selection` a matrix summarising the selection of eigenvectors for inclusion, with columns:

- **Step** Step counter of the selection procedure
- **SelEvec** number of selected eigenvector (sorted descending)
- **Eval** its associated eigenvalue
- **MinMi** value Moran's I for residual autocorrelation
- **ZMinMi** standardized value of Moran's I assuming a normal approximation
- **pr(ZI)** probability value of the permutation-based standardized deviate for the given value of the alternative argument
- **R2** R\^2 of the model including exogenous variables and eigenvectors
- **gamma** regression coefficient of selected eigenvector in fit

The first row is the value at the start of the search

`dataset` a matrix of the selected eigenvectors in order of selection

Author(s)

Yongwan Chun, Michael Tiefelsdorf, Roger Bivand

References

Tiefelsdorf M, Griffith DA. (2007) Semiparametric Filtering of Spatial Autocorrelation: The Eigenvector Approach. Environment and Planning A, 39 (5) 1193 - 1221. http://www.spatialfiltering.com

See Also

`lm`, `eigen`, `nb2listw`, `listw2U`

Examples

```
example(columbus)
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)
sarcol <- SpatialFiltering(CRIME ~ INC + HOVAL, data=columbus,
 nb=col.gal.nb, style="W", ExactEV=TRUE)
sarcol
lmsar <- lm(CRIME ~ INC + HOVAL + fitted(sarcol), data=columbus)
lmsar
anova(lmbase, lmsar)
lm.morantest(lmsar, nb2listw(col.gal.nb))
lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL - 1, data=columbus,
 nb=col.gal.nb, style="W")
lagcol
lmlag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
lmlag
anova(lmbase, lmlag)
lm.morantest(lmlag, nb2listw(col.gal.nb))
```

`spautolm` *Spatial conditional and simultaneous autoregression model estimation*

Description

Function taking family and weights arguments for spatial autoregression model estimation by Maximum Likelihood, using full matrix methods, not suited to large data sets with thousands of observations. With one of the sparse matrix methods, larger numbers of observations can be handled, but the `interval=` argument should be set. The implementation is GLS using the single spatial coefficient value, here termed lambda, found by line search using `optimize` to maximise the log likelihood.

Usage

```
spautolm(formula, data = list(), listw, weights,
 na.action, family = "SAR", method="full", verbose = NULL, interval=NULL,
 zero.policy = NULL, tol.solve=.Machine$double.eps, llprof=NULL,
 control=list())
## S3 method for class 'spautolm'
summary(object, correlation = FALSE, adj.se=FALSE,
 Nagelkerke=FALSE, ...)
```

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`

`weights`	an optional vector of weights to be used in the fitting process
`na.action`	a function (default `options("na.action")`), 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. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`family`	character string: either `"SAR"` or `"CAR"` for simultaneous or conditional autoregressions; `"SMA"` for spatial moving average added thanks to Jielai Ma - `"SMA"` is only implemented for method=`"full"` because it necessarily involves dense matrices
`method`	character string: default `"full"` for use of full matrices, `"Matrix_J"` or `"spam"` for sparse matrices (restricted to spatial weights symmetric or similar to symmetric) using methods in either the Matrix or spam packages; "Matrix" and "spam_update" provide updating Cholesky decomposition methods. Note that users should change from spam to Matrix. Values of method may also include "LU", which provides an alternative sparse matrix decomposition approach, and the "Chebyshev" and Monte Carlo "MC" approximate log-determinant methods.
`verbose`	default NULL, use global option value; if TRUE, reports function values during optimization.
`interval`	search interval for autoregressive parameter when not using method="eigen"; default is c(-1,0.999), `optimize` will reset NA/NaN to a bound and gives a warning when the interval is poorly set; method="Matrix" will attempt to search for an appropriate interval, if find_interval=TRUE (fails on some platforms)
`zero.policy`	default NULL, use global option value; Include list of no-neighbour observations in output if TRUE — otherwise zero.policy is handled within the listw argument
`tol.solve`	the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to `solve()` (default=double precision machine tolerance). Errors in `solve()` may constitute indications of poorly scaled variables: if the variables have scales differing much from the autoregressive coefficient, the values in this matrix may be very different in scale, and inverting such a matrix is analytically possible by definition, but numerically unstable; rescaling the RHS variables alleviates this better than setting tol.solve to a very small value
`llprof`	default NULL, can either be an integer, to divide the feasible range into llprof points, or a sequence of spatial coefficient values, at which to evaluate the likelihood function
`control`	list of extra control arguments - see section below
`object`	`spautolm` object from `spautolm`
`correlation`	logical; if 'TRUE', the correlation matrix of the estimated parameters is returned and printed (default=FALSE)
`adj.se`	if TRUE, adjust the coefficient standard errors for the number of fitted coefficients
`Nagelkerke`	if TRUE, the Nagelkerke pseudo R-squared is reported
`...`	further arguments passed to or from other methods

Details

This implementation is based on lm.gls and errorsarlm. In particular, the function does not (yet) prevent asymmetric spatial weights being used with "CAR" family models. It appears that both numerical issues (convergence in particular) and uncertainties about the exact spatial weights matrix used make it difficult to reproduce Cressie and Chan's 1989 results, also given in Cressie 1993.

Note that the fitted() function for the output object assumes that 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 (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data.

Value

A list object of class `spautolm`:

`fit`	a list, with items: **coefficients** ML coefficient estimates **SSE** ML sum of squared errors **s2** ML residual variance **imat** ML coefficient covariance matrix **signal_trend** non-spatial component of fitted.values **signal_stochastic** spatial component of fitted.values **fitted.values** sum of non-spatial and spatial components of fitted.values **residuals** difference between observed and fitted values
`lambda`	ML autoregressive coefficient
`LL`	log likelihood for fitted model
`LL0`	log likelihood for model with lambda=0
`call`	the call used to create this object
`parameters`	number of parameters estimated
`aliased`	if not NULL, details of aliased variables
`method`	Jacobian method chosen
`zero.policy`	zero.policy used
`weights`	case weights used
`interval`	the line search interval used
`timings`	processing timings
`na.action`	(possibly) named vector of excluded or omitted observations if non-default na.action argument used
`llprof`	if not NULL, a list with components lambda and ll of equal length

Control arguments

tol.opt: the desired accuracy of the optimization - passed to `optimize()` (default=`.Machine$double.eps^(2/3)`)

Imult: default 2; used for preparing the Cholesky decompositions for updating in the Jacobian function

super: if NULL (default), set to FALSE to use a simplicial decomposition for the sparse Cholesky decomposition and method "Matrix_J", set to `as.logical(NA)` for method "Matrix", if TRUE, use a supernodal decomposition

cheb_q: default 5; highest power of the approximating polynomial for the Chebyshev approximation

MC_p: default 16; number of random variates

MC_m: default 30; number of products of random variates matrix and spatial weights matrix

spamPivot: default "MMD", alternative "RCM"

in_coef default 0.1, coefficient value for initial Cholesky decomposition in "spam_update"

Note

The standard errors given in Waller and Gotway (2004) are adjusted for the numbers of parameters estimated, and may be reproduced by using the additional argument `adj.se=TRUE` in the `summary` method. In addition, the function returns fitted values and residuals as given by Cressie (1993) p. 564.

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; Waller, L. A., Gotway, C. A. 2004 *Applied spatial statistics for public health*, Wiley, Hoboken, NJ, 325-380; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York, 548-568; Ripley, B. D. 1981 *Spatial statistics*, Wiley, New York, 88-95; LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton.

See Also

`optimize`, `errorsarlm`, `do_ldet`

Examples

```
example(NY_data)
lm0 <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata)
summary(lm0)
lm0w <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, weights=POP8)
summary(lm0w)
esar0 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY)
```

```
summary(esar0)
system.time(esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="SAR", method="full", verbose=TRUE))
summary(esar1f)
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="SAR", method="Matrix", verbose=TRUE))
summary(esar1M)
## Not run:
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="SAR", method="Matrix", verbose=TRUE,
 control=list(super=TRUE)))
summary(esar1M)
system.time(esar1s <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="SAR", method="spam", verbose=TRUE))
summary(esar1s)
esar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="full")
summary(esar1wf)
system.time(esar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, weights=POP8, family="SAR", method="Matrix"))
summary(esar1wM)
esar1ws <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="spam")
summary(esar1ws)
esar1wlu <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="LU")
summary(esar1wlu)
esar1wch <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="Chebyshev")
summary(esar1wch)
ecar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, family="CAR", method="full")
summary(ecar1f)
system.time(ecar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="CAR", method="Matrix"))
summary(ecar1M)
ecar1s <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, family="CAR", method="spam")
summary(ecar1s)
ecar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=nydata$POP8, family="CAR", method="full")
summary(ecar1wf)
system.time(ecar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, weights=POP8, family="CAR", method="Matrix"))
summary(ecar1wM)
ecar1ws <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="CAR", method="spam")
summary(ecar1ws)
example(nc.sids)
ft.SID74 <- sqrt(1000)*(sqrt(nc.sids$SID74/nc.sids$BIR74) + sqrt((nc.sids$SID74+1)/nc.sids$B
lm_nc <- lm(ft.SID74 ~ 1)
sids.nhbr30 <- dnearneigh(cbind(nc.sids$east, nc.sids$north), 0, 30, row.names=row.names(nc.
sids.nhbr30.dist <- nbdists(sids.nhbr30, cbind(nc.sids$east, nc.sids$north))
```

```
sids.nhbr <- listw2sn(nb2listw(sids.nhbr30, glist=sids.nhbr30.dist, style="B", zero.policy=T
dij <- sids.nhbr[,3]
n <- nc.sids$BIR74
el1 <- min(dij)/dij
el2 <- sqrt(n[sids.nhbr$to]/n[sids.nhbr$from])
sids.nhbr$weights <- el1*el2
sids.nhbr.listw <- sn2listw(sids.nhbr)
both <- factor(paste(nc.sids$L_id, nc.sids$M_id, sep=":"))
ft.NWBIR74 <- sqrt(1000)*(sqrt(nc.sids$NWBIR74/nc.sids$BIR74) + sqrt((nc.sids$NWBIR74+1)/nc.
mdata <- data.frame(both, ft.NWBIR74, ft.SID74, BIR74=nc.sids$BIR74)
outl <- which.max(rstandard(lm_nc))
as.character(nc.sids$names[outl])
mdata.4 <- mdata[-outl,]
W <- listw2mat(sids.nhbr.listw)
W.4 <- W[-outl, -outl]
sids.nhbr.listw.4 <- mat2listw(W.4)
esarI <- errorsarlm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
 zero.policy=TRUE)
summary(esarI)
esarIa <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
 family="SAR")
summary(esarIa)
esarIV <- errorsarlm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbr.listw,
 zero.policy=TRUE)
summary(esarIV)
esarIVa <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbr.listw,
 family="SAR")
summary(esarIVa)
esarIaw <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
 weights=BIR74, family="SAR")
summary(esarIaw)
esarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata, listw=sids.nhbr.listw,
 weights=BIR74, family="SAR")
summary(esarIIaw)
esarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata,
 listw=sids.nhbr.listw, weights=BIR74, family="SAR")
summary(esarIVaw)
ecarIaw <- spautolm(ft.SID74 ~ 1, data=mdata.4, listw=sids.nhbr.listw.4,
 weights=BIR74, family="CAR")
summary(ecarIaw)
ecarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata.4,
 listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
summary(ecarIIaw)
ecarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata.4,
 listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
summary(ecarIVaw)
nc.sids$fitIV <- append(fitted.values(ecarIVaw), NA, outl-1)
spplot(nc.sids, c("fitIV"), cuts=12) # Cressie 1993, p. 565
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"))
summary(COL.errW.eig)
COL.errW.sar <- spautolm(CRIME ~ INC + HOVAL, data=COL.OLD,
```

```
 nb2listw(COL.nb, style="W"))
summary(COL.errW.sar)
data(boston)
gp1 <- spautolm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2)
 + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, nb2listw(boston.soi), family="SMA")
summary(gp1)

## End(Not run)
```

`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. Note: from spdep 0.3-32, the values of S1 and S2 are returned correctly for both underlying symmetric and asymmetric neighbour lists, before 0.3-32, S1 and S2 were wrong for listw objects based on asymmetric neighbour lists, such as k-nearest neighbours (thanks to Luc Anselin for finding the bug).

Usage

```
spweights.constants(listw, zero.policy=NULL, adjust.n=TRUE)
Szero(listw)
```

Arguments

listw	a listw object from for example nb2listw
zero.policy	default NULL, use global option value; if TRUE ignore zones without neighbours, if FALSE fail when encountered
adjust.n	default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

Value

n	number of zones
n1	n - 1
n2	n - 2
n3	n - 3
nn	n * n
S0	global sum of weights
S1	S1 sum of weights
S2	S2 sum of weights

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Haining, R. 1990 Spatial data analysis in the social and environmental sciences, Cambridge University Press, p. 233; Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 19, 21.

See Also

nb2listw

Examples

```
data(oldcol)
B <- spweights.constants(nb2listw(COL.nb, style="B"))
W <- spweights.constants(nb2listw(COL.nb, style="W"))
C <- spweights.constants(nb2listw(COL.nb, style="C"))
S <- spweights.constants(nb2listw(COL.nb, style="S"))
U <- spweights.constants(nb2listw(COL.nb, style="U"))
print(data.frame(rbind(unlist(B), unlist(W), unlist(C), unlist(S), unlist(U)),
  row.names=c("B", "W", "C", "S", "U")))
```

ssw *Compute the sum of dissimilarity*

Description

This function computes the sum of dissimilarity between each observation and the mean (scalar of vector) of the observations.

Usage

```
ssw(data, id, method = c("euclidean", "maximum",
    "manhattan", "canberra", "binary", "minkowski", "mahalanobis",
    "other"), p = 2, cov, inverted = FALSE, otherfun)
```

Arguments

data	A matrix with observations in the nodes.
id	Node index to compute the cost
method	Character for declare the distance method. For "euclidean", "maximum", "manhattan", "canberra", "binary" and "minkowski", see dist for details, because this function as used to compute the distance. If method="mahalanobis", the mahalanobis distance is computed between neighbour areas. If method="other", any function must be informed in otherfun argument.
p	The power of the Minkowski distance.
cov	The covariance matrix used to compute the mahalanobis distance.
inverted	logical. If 'TRUE', 'cov' is supposed to contain the inverse of the covariance matrix.
otherfun	A user defined function to compute the distance

Value

A numeric, the sum of dissimilarity between the observations id of data and the mean (scalar of vector) of this observations.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as nbcost

Examples

```
data(USArrests)
n <- nrow(USArrests)
ssw(USArrests, 1:n)
ssw(USArrests, 1:(n/2))
ssw(USArrests, (n/2+1):n)
ssw(USArrests, 1:(n/2)) + ssw(USArrests, (n/2+1):n)
```

`stsls` *Generalized spatial two stage least squares*

Description

The function fits a spatial lag model by two stage least squares, with the option of adjusting the results for heteroskedasticity.

Usage

```
stsls(formula, data = list(), listw, zero.policy = NULL,
 na.action = na.fail, robust = FALSE, legacy=FALSE, W2X = TRUE)
```

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`
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `lagsarlm()` to terminate with an error
`na.action`	a function (default `na.fail`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`robust`	default FALSE, if TRUE, apply a heteroskedasticity correction to the coefficients covariances
`legacy`	the argument chooses between two implementations of the robustness correction: default FALSE - use the estimate of Omega only in the White consistent estimator of the variance-covariance matrix, if TRUE, use the original implementation which runs a GLS using the estimate of Omega, and yields different coefficient estimates as well - see example below
`W2X`	default TRUE, if FALSE only WX are used as instruments in the spatial two stage least squares; until release 0.4-60, only WX were used - see example below

Details

The fitting implementation fits a spatial lag model:

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

by using spatially lagged X variables as instruments for the spatially lagged dependent variable.

Value

an object of class "stsls" containing:

`coefficients`	coefficient estimates
`var`	coefficient covariance matrix
`sse`	sum of squared errors
`residuals`	model residuals
`df`	degrees of freedom

Author(s)

Luc Anselin, Gianfranco Piras and Roger Bivand

References

Kelejian, H.H. and I.R. Prucha (1998). A generalized spatial two stage least squares procedure for estimating a spatial autoregressive model with autoregressive disturbances. *Journal of Real Estate Finance and Economics* 17, 99-121.

See Also

`lagsarlm`

Examples

```
data(oldcol)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
summary(COL.lag.eig, correlation=TRUE)
COL.lag.stsls <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
summary(COL.lag.stsls, correlation=TRUE)
COL.lag.stslsW <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb), W2X=FALSE)
summary(COL.lag.stslsW, correlation=TRUE)
COL.lag.stslsR <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb),
robust=TRUE, W2X=FALSE)
summary(COL.lag.stslsR, correlation=TRUE)
COL.lag.stslsRl <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb),
robust=TRUE, legacy=TRUE, W2X=FALSE)
summary(COL.lag.stslsRl, correlation=TRUE)
data(boston)
gp2a <- stsls(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) +
  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, nb2listw(boston.soi))
summary(gp2a)
```

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 = NULL, ...)
```

Arguments

`x`	an object of class `listw`
`subset`	logical expression
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors - passed through to `nb2listw`
`...`	generic function pass-through

Value

The function returns an object of class `listw` with component `style` the same as the input object, component `neighbours` a list of integer vectors containing neighbour region number ids (compacted to run from 1:number of regions in subset), and component `weights` as the weights computed for `neighbours` using `style`.

Author(s)

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

See Also

`nb2listw`, `subset.nb`

Examples

```
example(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

```
example(columbus)
coords <- coordinates(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, longlat = NULL, scale = 1, ...)
print.nb(x, ...)
summary.listw(object, coords, longlat, zero.policy = NULL,
 scale = 1, ...)
print.listw(x, zero.policy = NULL, ...)
```

Arguments

object	an object of class nb
coords	matrix of region point coordinates or a SpatialPoints object
longlat	TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if coords is a SpatialPoints object, the value is taken from the object itself
...	additional arguments affecting the output produced
x	an object of class nb
zero.policy	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets
scale	passed through to stem() for control of plot length

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

plot.nb

Examples

```
example(columbus)
coords <- coordinates(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

```
## S3 method for class 'sarlm'
summary(object, correlation = FALSE, Nagelkerke = FALSE, Hausman=FALSE, ...)
## S3 method for class 'sarlm'
print(x, ...)
## S3 method for class 'summary.sarlm'
print(x, digits = max(5, .Options$digits - 3),
signif.stars = FALSE, ...)
```

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)
`Nagelkerke`	if TRUE, the Nagelkerke pseudo R-squared is reported
`Hausman`	if TRUE, the results of the Hausman test for error models are reported
`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; Nagelkerke NJD (1991) A note on a general definition of the coefficient of determination. Biometrika 78: 691-692.

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")
summary(COL.mix.eig, correlation=TRUE, Nagelkerke=TRUE)
COL.mix.M <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb), type="mixed", method="Matrix")
summary(COL.mix.M, correlation=TRUE, Nagelkerke=TRUE)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="W"), method="eigen")
summary(COL.errW.eig, correlation=TRUE, Nagelkerke=TRUE, Hausman=TRUE)
```

---

tolerance.nb — *Function to construct edges based on a tolerance angle and a maximum distance*

Description

This function creates an object of class nb (defined in the library spdep) containing a connexion diagram. The edges between sites are based on a tolerance angle and a maximum distance. The angle is directional; its direction is always from the bottow to the top of the screen.

Usage

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

Arguments

coords	A matrix or a data frame containing the X and Y coordinates of the study sites.
unit.angle	Character. The measurement units in which angles are defined: either "degrees" (default) or "radians".

`max.dist`	Numeric. The maximum distance of an edge linking two sites together.
`tolerance`	Numeric. The tolerance angle in which a site can influence another site. The angle is measured vertically and from bottom to top of the pictures after rotation of the points.
`rot.angle`	Numeric, optional. An angle at which a set of coordinates should be rotated before creating the connexion diagram. The set of coordinates is rotated counterclockwise. Negative values will produce a clockwise rotation.
`plot.sites`	Logical (`TRUE`, `FALSE`) determining if the site should be plotted in a graphic window. This graph allows one to make sure the points are rotated in a correct direction.

Details

Even though this function creates a connexion diagram based on a tolerance angle going from the bottom to the top of the screen, the resulting object is symmetric, meaning that a site influences another and vice versa. The final object does not represent a directional connexion network.

Value

The function returns an object of class `nb` with a list of integer vectors corresponding to neighbour region numbers.

Warning

This function was not design to handle a large number of rows in `coords`. To use this function for a set of coordinates with more than 1500 entries is memory intensive.

Author(s)

F. Guillaume Blanchet

See Also

`dnearneigh`, `cell2nb`, `graph2nb`, `tri2nb`, `knn2nb`

Examples

```
set.seed(1)
ex.data<-cbind(runif(50),rexp(50))

### Construct object of class nb with a tolerance angle of 30 degrees and a maximum distance
nb.ex<-tolerance.nb(ex.data, unit.angle = "degrees", max.dist=1, tolerance = 30)

### Construct object of class nb with a tolerance angle of 30 degrees and a maximum distance
nb.ex2<-tolerance.nb(ex.data, unit.angle = "degrees", max.dist=1, tolerance = 30, rot.angle

### Construct object of class nb with a tolerance angle of pi/8 radians and a maximum distan
nb.ex3<-tolerance.nb(ex.data, unit.angle = "radians", max.dist=1.5, tolerance = pi/8,rot.ang
```

```
par(mfrow=c(1,3))
plot(nb.ex,ex.data,asp=1)
plot(nb.ex2,ex.data,asp=1)
plot(nb.ex3,ex.data,asp=1)
```

---

tri2nb *Neighbours list from tri object*

Description

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

Usage

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

Arguments

`coords`	matrix 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 coordinates are duplicated, this function cannot be used. If the coordinates are from a grid, then they need to be ordered such that the first three are not collinear, so that the first triangle can be constructed. This can be achieved by randomising the order of the coordinates (possibly several times), and then re-ordering the order of the data to match the new order of the neighbour list - if this fix is used, remember to re-order the row.names argument as well as the coordinates! Please also note that triangulation of grid points will give arbitrary diagonal neighbours, which may not be a sensible outcome, and `dnearneigh()` may serve better where `tri2nb()` cannot be used.

Value

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

Author(s)

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

See Also

`knn2nb`, `dnearneigh`, `cell2nb`

Examples

```
example(columbus)
coords <- coordinates(columbus)
ind <- sapply(slot(columbus, "polygons"), function(x) slot(x, "ID"))
col.tri.nb <- tri2nb(coords, row.names=ind)
plot(columbus, border="grey")
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,])
```

---

trW *Spatial weights matrix powers traces*

Description

The function is used to prepare a vector of traces of pwers of a spatial weights matrix

Usage

```
trW(W=NULL, m = 30, p = 16, type = "mult", listw=NULL)
```

Arguments

W	A spatial weights matrix in CsparseMatrix form
m	The number of powers; must be an even number for 'type'="moments" (default changed from 100 to 30 (2010-11-17))
p	The number of samples used in Monte Carlo simulation of the traces if type is MC (default changed from 50 to 16 (2010-11-17))
type	Either "mult" (default) for powering a sparse matrix (with moderate or larger N, the matrix becomes dense, and may lead to swapping), or "MC" for Monte Carlo simulation of the traces, or "moments" to use the looping space saving algorithm proposed by Smirnov and Anselin (2009) - for "moments", `W` must be symmetric, for row-standardised weights through a similarity transformation
listw	a listw object, which should either be fully symmetric, or be constructed as similar to symmetric from intrinsically symmetric neighbours using `similar.listw`, used with 'type'="moments"

Value

A numeric vector of `m` traces.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

LeSage J and RK Pace (2009) *Introduction to Spatial Econometrics*. CRC Press, Boca Raton, pp. 96–105; Smirnov O and L Anselin (2009) An O(N) parallel method of computing the Log-Jacobian of the variable transformation for models with spatial interaction on a lattice. *Computational Statistics and Data Analysis* 53 (2009) 2983–2984.

See Also

as_dgRMatrix_listw, nb2listw

Examples

```
example(columbus)
listw <- nb2listw(col.gal.nb)
W <- as(as_dgRMatrix_listw(listw), "CsparseMatrix")
system.time(trMat <- trW(W, type="mult"))
str(trMat)
set.seed(1)
system.time(trMC <- trW(W, type="MC"))
str(trMC)
plot(trMat, trMC)
listwS <- similar.listw(listw)
W <- as(as(as_dgRMatrix_listw(listwS), "CsparseMatrix"), "symmetricMatrix")
system.time(trmom <- trW(W, m=24, type="moments"))
str(trmom)
all.equal(trMat[1:24], trmom, check.attributes=FALSE)
system.time(trMat <- trW(W, m=24, type="mult"))
str(trMat)
all.equal(trMat, trmom, check.attributes=FALSE)
set.seed(1)
system.time(trMC <- trW(W, m=24, type="MC"))
str(trMC)
```

---

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(used.cars$price.1960, nb2listw(usa48.nb))
moran.plot(used.cars$price.1960, 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, control=list(tol.opt=.Machine$double.eps^0.3))
summary(uc.err)
uc.lag <- lagsarlm(price.1960 ~ tax.charges, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, control=list(tol.opt=.Machine$double.eps^0.3))
summary(uc.lag)
uc.lag1 <- lagsarlm(price.1960 ~ 1, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, control=list(tol.opt=.Machine$double.eps^0.3))
summary(uc.lag1)
uc.err1 <- errorsarlm(price.1960 ~ 1, data=used.cars,
  nb2listw(usa48.nb), tol.solve=1.0e-13, control=list(tol.opt=.Machine$double.eps^0.3))
summary(uc.err1)
```

`wheat` *Mercer and Hall wheat yield data*

Description

Mercer and Hall wheat yield data, based on version in Cressie (1993), p. 455.

Usage

```
data(wheat)
```

Format

The format of the object generated by running `data(wheat)` is a three column data frame made available by Hongfei Li. The example section shows how to convert this to the object used in demonstrating the `aple` function, and is a: Formal class 'SpatialPolygonsDataFrame' [package "sp"] with 5 slots; the data slot is a data frame with 500 observations on the following 6 variables.

- `lat` local coordinates northings ordered north to south
- `yield` Mercer and Hall wheat yield data
- `r` rows south to north; levels in distance units of plot centres
- `c` columns west to east; levels in distance units of plot centres
- `lon` local coordinates eastings
- `lat1` local coordinates northings ordered south to north

Source

Cressie, N. A. C. (1993) Statistics for Spatial Data. Wiley, New York, p. 455.

References

Mercer, W. B. and Hall, A. D. (1911) The experimental error of field trials. Journal of Agricultural Science 4, 107-132.

Examples

```
## Not run:
data(wheat)
names(wheat) <- c('lon','lat','yield')
wheat$lat1 <- 69 - wheat$lat
wheat$r <- factor(wheat$lat1)
wheat$c <- factor(wheat$lon)
wheat_sp <- wheat
coordinates(wheat_sp) <- c("lon", "lat1")
wheat_spg <- wheat_sp
gridded(wheat_spg) <- TRUE
wheat_spl <- as(wheat_spg, "SpatialPolygons")
df <- as(wheat_spg, "data.frame")
row.names(df) <- sapply(slot(wheat_spl, "polygons"),
 function(x) slot(x, "ID"))
wheat <- SpatialPolygonsDataFrame(wheat_spl, data=df)

## End(Not run)
wheat <- readShapeSpatial(system.file("etc/shapes/wheat.shp",
 package="spdep")[1])
```

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

```
example(columbus)
GALfile <- tempfile("GAL")
write.nb.gal(col.gal.nb, GALfile)
col.queen <- read.gal(GALfile)
summary(diffnb(col.queen, col.gal.nb))
```

Index