Package ‘spdep’

December 14, 2021

Version 1.1-13
Date 2021-12-13
Title Spatial Dependence: Weighting Schemes, Statistics
Encoding UTF-8
Depends R (>= 3.3.0), methods, sp (>= 1.0), spData (>= 0.2.6.0), sf
Imports deldir, boot (>= 1.3-1), graphics, grDevices, utils, stats, Matrix, LearnBayes, MASS, coda, expm, gmodels, nlme, units, s2, e1071
Suggests spatialreg (>= 1.1-1), parallel, dbscan, rgeos, RColorBrewer, lattice, xtable, maptools (>= 0.5-4), foreign, igraph, RSpectra, knitr, rgdal, classInt, tmap, spam, ggplot2, markdown, tinytest, rgeoda
BugReports https://github.com/r-spatial/spdep/issues/

and exact tests for global and local 'Moran's I' ('Bivand et al.' 2009) and 'LOSH' local indicators of spatial heteroscedasticity ('Ord' and 'Getis')<doi:10.1007/s00168-011-0492-y>. The implementation of most of the measures is described in 'Bivand' and 'Wong' (2018) <doi:10.1007/s11749-018-0599-x>.

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

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

License GPL (>= 2)

VignetteBuilder knitr

RoxygenNote RoxygenNote: 6.1.1

NeedsCompilation yes

Author Roger Bivand [cre, aut] (<https://orcid.org/0000-0003-2392-6140>), Micah Altman [ctb], Luc Anselin [ctb], Renato Assunção [ctb], Olaf Berke [ctb], Andrew Bernat [ctb], Guillaume Blanchet [ctb], Eric Blankmeyer [ctb], Marilia Carvalho [ctb], Bjarke Christensen [ctb], Yongwan Chun [ctb], Carsten Dormann [ctb], Stéphane Dray [ctb], Virgilio Gómez-Rubio [ctb], Martin Gubri [ctb], Rein Halbersma [ctb], Elias Krainski [ctb], Pierre Legendre [ctb], Nicholas Lewin-Koh [ctb], Angela Li [ctb], Hongfei Li [ctb], Jielai Ma [ctb], Abhirup Mallik [ctb, trl], Giovanni Millo [ctb], Werner Mueller [ctb], Hisaji Ono [ctb], Josiah Parry [ctb] (<https://orcid.org/0000-0001-9910-865X>), Pedro Peres-Neto [ctb],
R topics documented:

Gianfranco Piras [ctb],
Markus Reder [ctb],
Jeff Sauer [ctb],
Michael Tiefelsdorf [ctb],
René Westerholt [ctb],
Levi Wolf [ctb],
Danlin Yu [ctb]

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

Repository CRAN

Date/Publication 2021-12-14 13:00:02 UTC

R topics documented:

aggregate.nb .................................................. 5
airdist .............................................................. 6
ape ................................................................. 7
ape.mc .............................................................. 8
ape.plot ........................................................... 10
autocov_dist ...................................................... 11
bhicv ............................................................... 13
card ............................................................... 14
cell2nb ........................................................... 15
choynowski ....................................................... 17
columbus ......................................................... 18
diffnb ............................................................. 19
dnearnbh .......................................................... 20
droplinks ......................................................... 22
EBest .............................................................. 24
EBImoran.mc ...................................................... 25
EBlocal ........................................................... 27
edit.nb ............................................................ 29
eire ................................................................. 30
geary ............................................................... 30
geary.mc .......................................................... 31
geary.test ........................................................ 33
globalG.test ....................................................... 35
Graph Components ............................................... 36
graphneigh ....................................................... 38
grid2nb ........................................................... 41
include.self ...................................................... 42
invIrM ............................................................. 43
is.symmetric.nb .................................................. 46
joincount.mc ...................................................... 47
joincount.multi .................................................. 48
joincount.test .................................................... 50
knearnbh .......................................................... 52
knn2nb ........................................................... 54
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>lag.listw</td>
<td>55</td>
</tr>
<tr>
<td>lee</td>
<td>56</td>
</tr>
<tr>
<td>lee.mc</td>
<td>57</td>
</tr>
<tr>
<td>lee.test</td>
<td>59</td>
</tr>
<tr>
<td>listw2sn</td>
<td>61</td>
</tr>
<tr>
<td>lm.LMtests</td>
<td>62</td>
</tr>
<tr>
<td>lm.morantest</td>
<td>64</td>
</tr>
<tr>
<td>lm.morantest.exact</td>
<td>66</td>
</tr>
<tr>
<td>lm.morantest.sad</td>
<td>68</td>
</tr>
<tr>
<td>localC</td>
<td>70</td>
</tr>
<tr>
<td>localG</td>
<td>73</td>
</tr>
<tr>
<td>localmoran</td>
<td>75</td>
</tr>
<tr>
<td>localmoran.exact</td>
<td>78</td>
</tr>
<tr>
<td>localmoran.sad</td>
<td>81</td>
</tr>
<tr>
<td>LOSH</td>
<td>84</td>
</tr>
<tr>
<td>LOSH.cs</td>
<td>86</td>
</tr>
<tr>
<td>LOSH.mc</td>
<td>87</td>
</tr>
<tr>
<td>mat2listw</td>
<td>89</td>
</tr>
<tr>
<td>moran</td>
<td>90</td>
</tr>
<tr>
<td>moran.mc</td>
<td>91</td>
</tr>
<tr>
<td>moran.plot</td>
<td>93</td>
</tr>
<tr>
<td>moran.test</td>
<td>95</td>
</tr>
<tr>
<td>mstree</td>
<td>97</td>
</tr>
<tr>
<td>nb.set.operations</td>
<td>99</td>
</tr>
<tr>
<td>nb2blocknb</td>
<td>100</td>
</tr>
<tr>
<td>nb2INLA</td>
<td>102</td>
</tr>
<tr>
<td>nb2lines</td>
<td>103</td>
</tr>
<tr>
<td>nb2listw</td>
<td>104</td>
</tr>
<tr>
<td>nb2listwdist</td>
<td>106</td>
</tr>
<tr>
<td>nb2mat</td>
<td>109</td>
</tr>
<tr>
<td>nb2WB</td>
<td>110</td>
</tr>
<tr>
<td>nbcosts</td>
<td>111</td>
</tr>
<tr>
<td>nbdists</td>
<td>112</td>
</tr>
<tr>
<td>nblag</td>
<td>113</td>
</tr>
<tr>
<td>oldcol</td>
<td>115</td>
</tr>
<tr>
<td>p.adjustSP</td>
<td>116</td>
</tr>
<tr>
<td>plot.mst</td>
<td>117</td>
</tr>
<tr>
<td>plot.nb</td>
<td>118</td>
</tr>
<tr>
<td>plot.skater</td>
<td>119</td>
</tr>
<tr>
<td>poly2nb</td>
<td>120</td>
</tr>
<tr>
<td>probmap</td>
<td>122</td>
</tr>
<tr>
<td>prunecost</td>
<td>123</td>
</tr>
<tr>
<td>prunemst</td>
<td>125</td>
</tr>
<tr>
<td>read.gal</td>
<td>126</td>
</tr>
<tr>
<td>read.gwt2nb</td>
<td>127</td>
</tr>
<tr>
<td>Rotation</td>
<td>129</td>
</tr>
<tr>
<td>set.mcOption</td>
<td>130</td>
</tr>
<tr>
<td>set.spChkOption</td>
<td>132</td>
</tr>
</tbody>
</table>
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

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

Arguments

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

Value

an nb neighbour object, with empty aggregates dropped.

Note

Method suggested by Roberto Patuelli

Author(s)

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

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

---

`airdist`  
Measure distance from plot

Description

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

Usage

`airdist(ann=FALSE)`

Arguments

- `ann` annotate the plot with line measured and distance

Value

A list with members:

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

- `locator`
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
apel(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 \(W \times W\) (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
See Also

nb2listw, aple.mc, aple.plot

Examples

```r
wheat <- st_read(system.file("shapes/wheat.shp", package="spData")[[1]], quiet=TRUE)
nbr1 <- poly2nb(wheat, queen=FALSE)
nbrl <- nblag(nbr1, 2)
nbr12 <- nblag_cumul(nbr1)
cms0 <- with(as.data.frame(wheat), tapply(yield, c, median))
cms1 <- c(model.matrix(~ factor(c) -1, data=wheat) %*% cms0)
wheat$yield_detrend <- wheat$yield - cms1
isTRUE(all.equal(c(with(as.data.frame(wheat),
tapply(yield_detrend, c, median))), rep(0.0, 25),
check.attributes=FALSE))
moran.test(wheat$yield_detrend, nb2listw(nbr12, style="W"))
aple(as.vector(scale(wheat$yield_detrend, scale=FALSE)), nb2listw(nbr12, style="W"))
```

```
## Not run:
errorsarlm(yield_detrend ~ 1, wheat, nb2listw(nbr12, style="W"))
## End(Not run)
```

ape.mc

Approximate profile-likelihood estimator (APLE) permutation test

Description

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

Usage

```r
ape.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

See Also
aple.boot

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

Distance-weighted autocovariate

Description

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

References


See Also

aple

Examples

```r
## Not run:
wheat <- st_read(system.file("shapes/wheat.shp", package="spData")[[1]], quiet=TRUE)
nbr1 <- poly2nb(wheat, queen=FALSE)
nbrl <- nblag(nbr1, 2)

## End(Not run)
```
Usage

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

Arguments

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

Value

A numeric vector of autocovariate values

Note

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

Author(s)

Carsten F. Dormann and Roger Bivand

References


See Also

`nb2listw`
Examples

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
#xy <- cbind(columbus$X, columbus$Y)
xy <- st_coordinates(st_centroid(st_geometry(columbus),
of_largest_polygon=TRUE))
acla <- autocov_dist(columbus$CRIME, xy, nbs=10, style="B",
type="one")
acinva <- autocov_dist(columbus$CRIME, xy, nbs=10, style="B",
type="inverse")
acinva2 <- autocov_dist(columbus$CRIME, xy, nbs=10, style="B",
type="inverse.squared")
plot(ac1a ~ columbus$CRIME, pch=16, ylim=c(0,9000))
points(acinva ~ columbus$CRIME, pch=16, col="red")
points(acinv2a ~ columbus$CRIME, pch=16, col="blue")
legend("topleft", legend=c("one", "inverse", "inverse.squared"),
col=c("black", "red", "blue"), bty="n", pch=16)
nb <- dnearneigh(xy, 0, 10)
lw <- nb2listw(nb, style="B")
ac1b <- lag(lw, columbus$CRIME)
all.equal(ac1b, ac1a)

nb <- nb2listw(nb, style="B")
gl <- lapply(nb, function(x) 1/x)
lw <- nb2listw(nb, glist=gl, style="B")
acinvb <- lag(lw, columbus$CRIME)
all.equal(acinvb, acinva)
gl2 <- lapply(nb, function(x) 1/(x^2))
lw <- nb2listw(nb, glist=gl2, style="B")
acinvb2 <- lag(lw, columbus$CRIME)
all.equal(acinvb2, acinv2a)

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

bhicv

Data set with 4 life condition indices of Belo Horizonte region

description

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

A shape polygon object with seven variables:

- **id**  The identifier
- **Name**  Name of city
- **Population**  The population of city
- **HLCI**  Health Life Condition Index
- **ELCI**  Education Life Condition Index
- **CLCI**  Children Life Condition Index
- **ELCI**  Economic Life Condition Index

Examples

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

---

**card**

*Cardinalities for neighbours lists*

Description

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

Usage

```r
card(nb)
```

Arguments

- **nb**  a neighbours list object of class nb

Details

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

Value

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

Author(s)

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


See Also

summary.nb

Examples

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

cell2nb

Generate neighbours list for grid cells

Description

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

Usage

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

Arguments

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

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`summary.nb`, `card`

Examples

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

Choynowski probability map values

Description

Calculates Choynowski probability map values.

Usage

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

Arguments

n  
a numeric vector of counts of cases

x  
a numeric vector of populations at risk

row.names  
row names passed through to output data frame

tol  
accumulate values for observed counts >= expected until value less than tol

legacy  
default FALSE using vectorised alternating side ppois version, if true use original version written from sources and iterating down to tol

Value

A data frame with columns:

- **pmap**: Poisson probability map values: probability 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


See Also

probmap
Examples

```r
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
columbus <- st_read(system.file("shapes/auckland.shp", package="spData")[1], quiet=TRUE)
auckland.nb <- poly2nb(auckland)
res <- choynowski(auckland$M77_85, 9*auckland$Und5_81)
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)
side <- c("greater", "less")
for (i in seq(along=ch_ppois_pmap)) {
  ch_ppois_pmap[i] <- poisson.test(auckland$M77_85[i], r=rt,
                                 T=(9*auckland$Und5_81[i]), alternative=side[(res$type[i]+1)])$p.value
}
all.equal(ch_ppois_pmap, res$pmap)
res1 <- probmap(auckland$M77_85, 9*auckland$Und5_81)
table(abs(res$pmap - res1$pmap) < 0.00001, res$type)
lt005 <- (res$pmap < 0.05) & (res$type)
ge005 <- (res$pmap < 0.05) & (!res$type)
cols <- rep("nonsig", length(lt005))
cols[lt005] <- "low"
cols[ge005] <- "high"
auckland$cols <- factor(cols)
plot(auckland[,"cols"], main="Probability map")
```

columbus

*Columbus OH spatial analysis data set*

Description

The data set is now part of the spData package

Usage

```r
data(columbus)
```

Examples

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

Differences between neighbours lists

Description

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

Usage

```r
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

```r
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)n <- row.names(columbus)
knn1 <- knearneigh(coords, 1)
knn2 <- knearneigh(coords, 2)
b1 <- knn2nb(knn1, row.names=r)
b2 <- knn2nb(knn2, row.names=r)
diffs <- diffnb(b2, b1)
opar <- par(no.readonly=TRUE)
plot(st_geometry(columbus), border="grey", reset=FALSE,
     main="Plot of first (black) and second (red)\nnearest neighbours")
plot(b1, coords, add=TRUE)
plot(diffs, coords, add=TRUE, col="red", lty=2)
par(opar)
```
Description

The function identifies neighbours of region points by Euclidean distance between lower (greater than or equal to (changed from version 1.1-7)) and upper (less than or equal to) bounds, or with longlat = TRUE, by Great Circle distance in kilometers.

Usage

dnearneigh(x, d1, d2, row.names = NULL, longlat = NULL, bounds=c("GE", "LE"), use_kd_tree=TRUE, symtest=FALSE, use_s2=FALSE, max_cells=200, dwithin=FALSE)

Arguments

x matrix of point coordinates, an object inheriting from SpatialPoints or an "sf" or "sfc" object; if the "sf" or "sfc" object geometries are in geographical coordinates (use_s2=FALSE, sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE), s2 will be used to find the neighbours because it will (we hope) use spatial indexing https://github.com/r-spatial/s2/issues/125 as opposed to the legacy method which uses brute-force (at present s2 also uses brute-force)

d1 lower distance bound

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, and overrides this argument if not NULL

bounds character vector of length 2, default c("GE", "LE"). (GE: greater than or equal to, LE: less than or equal to) that is the finite and closed interval \([d1,d2]\), \(d1 \leq x \leq d2\). The first element may also be "GT" (GT: greater than), the second "LT" (LT: less than) for finite, open intervals excluding the bounds; the first bound default was changed from "GT" to "GE" in release 1.1-7. When creating multiple distance bands, finite, half-open right-closed intervals may be used until the final interval to avoid overlapping on bounds: "GE", "LT", that is \([d1,d2)\), \(d1 \leq x < d2\)

use_kd_tree default TRUE, if TRUE, use dbscan frNN if available (permitting 3D distances).

symtest Default FALSE; before release 1.1-7, TRUE - run symmetry check on output object, costly with large numbers of points.

use_s2 default=FALSE, as of s2 1.0-5, distance bound computations do not use spatial indexing so even if sf::sf_use_s2() is TRUE, s2 will not be used for distances on the sphere for "sf" or "sfc" objects; the default will change when/if spatial
dnearneigh

indexing becomes available. Until then, setting set.coresOption to a positive integer will experimentally use that number of cores on .Platform$OS.type== "unix" systems.

max_cells  default 200; if use_s2=TRUE, sf::st_is_longlat(x) == TRUE, sf::sf_use_s2() == TRUE and dwithin=FALSE, set max_cells in s2::s2_buffer_cells()

dwithin  default FALSE to use s2::s2_buffer_cells() and s2::s2_intersects_matrix(), if TRUE, use s2::s2_dwithin_matrix(), both if use_s2=TRUE, sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE; both yield the same lists of neighbours.

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

kneareigh, card

Examples

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
rn <- row.names(columbus)
k1 <- knn2nb(knearneigh(coords))
all.linked <- max(unlist(nbdists(k1, coords)))
col.nb.0.all <- dnearneigh(coords, 0, all.linked, row.names=rn)
s2::s2_buffer_cells()

opar <- par(no.readonly=TRUE)
plot(st_geometry(columbus), border="grey", reset=FALSE,
main=paste("Distance based neighbours 0-", format(all.linked), sep=""))
plot(col.nb.0.all, coords, add=TRUE)
par(opar)

(sfc_obj <- st_centroid(st_geometry(columbus)))
col.nb.0.all_sf <- dnearneigh(sfc_obj, 0, all.linked, row.names=rn)
all.equal(col.nb.0.all, col.nb.0.all_sf, check.attributes=FALSE)

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))))
droplinks

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 an 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


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)
**Description**

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

**Usage**

```r
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
EBImoran.mc

References


See Also

EBlocal, probmap, EBImoran.mc

Examples

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

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

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

Arguments

- `n`: a numeric vector of counts of cases the same length as the neighbours list in `listw`
- `x`: a numeric vector of populations at risk the same length as the neighbours list in `listw`
- `listw`: a `listw` object created for example by `nb2listw`
- `nsim`: number of permutations
zero.policy default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative a character string specifying the alternative hypothesis, must be one of "greater" (default), or "less"
spChk should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
return_boot return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest
subtract_mean_in_numerator default TRUE, if TRUE subtract mean of z in numerator of EBI equation on p. 2157 in reference (consulted with Renato Assunção 2016-02-19); until February 2016 the default was FALSE agreeing with the printed paper.

Details

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

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

where:

$$z_{i} = \frac{p_{i} - b}{\sqrt{v_{i}}}$$

and:

$$p_{i} = \frac{n_{i}}{x_{i}}$$

$$v_{i} = a + \left( \frac{b}{x_{i}} \right)$$

$$b = \sum_{i=1}^{m} \frac{n_{i}}{\sum_{i=1}^{m} x_{i}}$$

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

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

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
EBlocal

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
moran, moran.mc, EBest

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

---

EBlocal

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


See Also

EBest, probmap

Examples

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

Description

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

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

Usage

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

Arguments

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

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

- `summary.nb`
- `plot.nb`

Examples

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

**Eire data sets**

**Description**

The data set is now part of the spData package

**Usage**

```r
data(eire)
```

geary  

**Compute Geary's C**

**Description**

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

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

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

**Usage**

```r
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

See Also
geary.test, geary.mc, sp.mantel.mc

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

geary.mc
Permutation test for Geary's C statistic

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

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

Arguments
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"; this reversal corresponds to that on geary.test described
in the section on the output statistic value, based on Cliff and Ord 1973, p. 21
(changed 2011-04-11, thanks to Daniel Garavito).
spChk  should the data vector names be checked against the spatial objects for identity
integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.n: default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

return_boot: return an object of class boot from the equivalent permutation bootstrap rather than an object of class htest

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


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'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**

```r
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


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)
Global G test for spatial autocorrelation

Description

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

Usage

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

Arguments

x               a numeric vector the same length as the neighbours list in listw
listw           a listw object created for example by nb2listw; if a sequence of distance bands is to be used, it is recommended that the weights style be binary (one of c("B","C","U").
zero.policy     default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
alternative     a character string specifying the alternative hypothesis, must be one of "greater" (default), "less" or "two.sided".
spChk           should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
adjust.n        default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
B1correct       default TRUE, if TRUE, the erratum referenced below: "On page 195, the coefficient of W2 in B1, (just below center of the page) should be 6, not 3." is applied; if FALSE, 3 is used (as in CrimeStat IV)
adjust.x        default TRUE, if TRUE, x values of observations with no neighbours are omitted in the denominator of G
Arc_all_x       default FALSE, if Arc_all_x=TRUE and adjust.x=TRUE, use the full x vector in part of the denominator term for G

Value

A list with class htest containing the following components:

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

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

References


See Also

localG

Examples

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

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)
Graph Components

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

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

Graph based spatial weights

Description

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

Usage

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

soi.graph(tri.nb, coords, quadsegs=10)
graph2nb(gob, row.names=NULL, sym=FALSE)

## S3 method for class 'Gabriel'
plot(x, show.points=FALSE, add=FALSE, linecol=par(col), ...)
## S3 method for class 'relative'
plot(x, show.points=FALSE, add=FALSE, linecol=par(col), ...)

Arguments

coords matrix of region point coordinates or SpatialPoints object or sfc points object
nnmult scaling factor for memory allocation, default 3; if higher values are required, the function will exit with an error; example below thanks to Dan Putler
tri.nb a neighbor list created from tri2nb
quadsegs number of line segments making a quarter circle buffer, see the nQuadSegs argument in geos_unary
gob a graph object created from any of the graph functions
row.names character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1,nrow(x))
sym a logical argument indicating whether or not neighbors should be symmetric (if i->j then j->i)
x object to be plotted
show.points (logical) add points to plot
add (logical) add to existing plot
linecol edge plotting colour
... further graphical parameters as in par(..)
Details

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

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

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

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

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

See \texttt{card} for details of “nb” objects.

Value

A list of class \texttt{Graph} with the following elements

- \texttt{np} number of input points
- \texttt{from} array of origin ids
- \texttt{to} array of destination ids
- \texttt{nedges} number of edges in graph
- \texttt{x} input x coordinates
- \texttt{y} input y coordinates

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

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

References


See Also

knearneigh, dnearneigh, knn2nb, card

Examples

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
sf_obj <- st_centroid(st_geometry(columbus), of_largest_polygon)
sp_obj <- as(sf_obj, "Spatial")
coords <- st_coordinates(sf_obj)
suppressMessages(col.tri.nb <- tri2nb(coords))
col.gab.nb <- graph2nb(gabrielneigh(coords), sym=TRUE)
col.rel.nb <- graph2nb(relativeneigh(coords), sym=TRUE)
par(mfrow=c(2,2))
plot(st_geometry(columbus), border="grey")
plot(col.tri.nb, coords, add=TRUE)
title(main="Delaunay Triangulation", cex.main=0.6)
plot(st_geometry(columbus), border="grey")
plot(col.gab.nb, coords, add=TRUE)
title(main="Gabriel Graph", cex.main=0.6)
plot(st_geometry(columbus), border="grey")
plot(col.rel.nb, coords, add=TRUE)
title(main="Relative Neighbor Graph", cex.main=0.6)
if (require("dbscan", quietly=TRUE)) {
col.soi.nb <- graph2nb(soi.graph(col.tri.nb, coords), sym=TRUE)
plot(col.soi.nb, coords, add=TRUE)
title(main="Sphere of Influence Graph", cex.main=0.6)
}
par(mfrow=c(1,1))
col.tri.nb.sf <- tri2nb(sf_obj)
all.equal(col.tri.nb, col.tri.nb.sf, check.attributes=FALSE)
col.tri.nb.sp <- tri2nb(sp_obj)
all.equal(col.tri.nb, col.tri.nb_sp, check.attributes=FALSE)
if (require("dbscan", quietly=TRUE)) {
col.soi.nb.sf <- graph2nb(soi.graph(col.tri.nb, sf_obj), sym=TRUE)
all.equal(col.soi.nb, col.soi.nb.sf, check.attributes=FALSE)
col.soi.nb_sp <- graph2nb(soi.graph(col.tri.nb, sp_obj), sym=TRUE)
all.equal(col.soi.nb, col.soi.nb_sp, check.attributes=FALSE)
}
col.gab.nb_sp <- graph2nb(gabrielneigh(sp_obj), sym=TRUE)
all.equal(col.gab.nb, col.gab.nb_sp, check.attributes=FALSE)
col.gab.nb.sf <- graph2nb(gabrielneigh(sf_obj), sym=TRUE)
all.equal(col.gab.nb, col.gab.nb_sf, check.attributes=FALSE)
col.rel.nb_sp <- graph2nb(relativeneigh(sp_obj), sym=TRUE)
all.equal(col.rel.nb, col.rel.nb_sp, check.attributes=FALSE)
col.rel.nb.sf <- graph2nb(relativeneigh(sf_obj), sym=TRUE)
all.equal(col.rel.nb, col.rel.nb_sf, check.attributes=FALSE)
dx <- rep(0.25*0:4,5)
dy <- c(rep(0,5),rep(0.25,5),rep(0.5,5), rep(0,5,5), rep(0.25,5),rep(1,5))
m <- cbind(dx, dy, 3+dx, 3+dy)
cat(try(res <- gabrielneigh(m), silent=TRUE), "\n")
res <- gabrielneigh(m, nnnmult=4)
grid2nb

Construct neighbours for a GridTopology

Description

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

Usage

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

Arguments

grid An object of class GridTopology. One can avoid to supply this by just supplying the dimensions in the d argument.

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

queen Logical. Default is TRUE. To inform if the queen neighbourhood structure should be considered. If FALSE, only a hyper-cube with a common face will be considered neighbour. If TRUE, a single shared coordinate meets the contiguity condition.

nb Default TRUE. If TRUE, return the result as a neighbours list with class nb. If FALSE, the result is a matrix with \( 3^k \) columns if self = TRUE or \( 3^k-1 \) if self = FALSE. Zeros are used for hyper-cubes at boundaries.

self Default FALSE, to indicate if the hyper-cube neighbour itself should be considered a neighbour.

Value

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

Note

This applies to a k-dimensional grid topology.
Author(s)
Elias T Krainski <eliaskrainski@gmail.com>

See Also
poly2nb, summary.nb, card

Examples

```r
nb <- grid2nb(d = c(5L, 5L, 5L))
nb
summary(nb)

gt <- GridTopology(c(.125,.1), c(.25,.2), c(4L, 5L))
nb1 <- grid2nb(gt, queen = FALSE)
nb2 <- grid2nb(gt)

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

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

include.self

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>
invIrM  

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

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

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

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

\[
(I - \rho V)^{-1}
\]
. With method="chol" (only for a listw object), Cholesky decomposition is used, thanks to contributed code by Markus Reder and Werner Mueller.

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

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

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

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

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

from

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

, 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, but is sensitive to the power weights order and the tolerance arguments when the spatial coefficient is close to its bounds, leading to incorrect estimates of the implied inverse matrix.

Value

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

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

.

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

invIrM

See Also

nb2listw

Examples

nb7rt <- cell2nb(7, 7, torus=TRUE)
lw <- nb2listw(nb7rt, style="W")
set.seed(1)
x <- matrix(sample(rnorm(500*length(nb7rt))), nrow=length(nb7rt))
if (requireNamespace("spatialreg", quietly=TRUE)) {
  # Only needed in some simulation settings where the input and
  # output distributions must agree in all but autocorrelation
  if (FALSE) {
    e <- spatialreg::eigenw(lw)
x <- apply(x, 2, scale)
st <- apply(x, 2, function(x) shapiro.test(x)$p.value)
x <- x[, (st > 0.2 & st < 0.8)]
x <- apply(x, 2, function(v) spatialreg::residuals.spautolm(    
      spatialreg::spautolm(v ~ 1, listw=lw, method="eigen",
      control=list(pre_eig=e, fdHess=FALSE)))
  x <- apply(x, 2, scale)
  }
res0 <- apply(invIrM(nb7rt, rho=0.0, method="chol", feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res2 <- apply(invIrM(nb7rt, rho=0.2, method="chol", feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res4 <- apply(invIrM(nb7rt, rho=0.4, method="chol", feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res6 <- apply(invIrM(nb7rt, rho=0.6, method="chol", feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res8 <- apply(invIrM(nb7rt, rho=0.8, method="chol", feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
res9 <- apply(invIrM(nb7rt, rho=0.9, method="chol", feasible=TRUE) %*% x, 2, function(x) var(x)/length(x))
plot(density(res9), col="red", xlim=c(-0.01, max(density(res9)$x)),
     ylim=range(density(res0)$y),
     xlab="estimated variance of the mean",
     main=expression(paste("Effects of spatial autocorrelation for different ",
       rho", " values")))
lines(density(res0), col="black")
lines(density(res2), col="brown")
lines(density(res4), col="green")
lines(density(res6), col="orange")
lines(density(res8), col="pink")
legend(c(-0.02, 0.01), c(7, 25),
       legend=c("0.0", "0.2", "0.4", "0.6", "0.8", "0.9"),
       col=c("black", "brown", "green", "orange", "pink", "red"), lty=1, bty="n")
}
## Not run:
x <- matrix(rnorm(length(nb7rt)), ncol=1)
/system.time(e <- invIrM(nb7rt, rho=0.9, method="chol", feasible=TRUE) %*% x)
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>
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


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"))

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.
joincount.multi

Usage

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

Arguments

fx           a factor of the same length as the neighbours and weights objects in listw
listw        a listw object created for example by nb2listw
zero.policy  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


See Also

joincount.test
joincount.test

**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**

```r
joincount.test(fx, listw, zero.policy=NULL, alternative="greater", sampling="nonfree", spChk=NULL, adjust.n=TRUE)
```

## S3 method for class 'jclist'

```r
print(x, ...)
```

**Arguments**

- `fx` a factor of the same length as the neighbours and weights objects in listw

**Examples**

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

## Not run:
data(hopkins, package="spData")
image(1:32, 1:32, hopkins[5:36,36:5], breaks=c(-0.5, 3.5, 20),
     col=c("white", "black"))
box()
hopkins.rook.nb <- cell2nb(32, 32, type="rook")
unlist(spweights.constants(nb2listw(hopkins.rook.nb, style="B")))
hopkins.queen.nb <- cell2nb(32, 32, type="queen")
hopkins.bishop.nb <- diffnb(hopkins.rook.nb, hopkins.queen.nb, verbose=FALSE)
hopkins4 <- hopkins[5:36,36:5]
hopkins4[which(hopkins4 > 3, arr.ind=TRUE)] <- 4
hopkins4.f <- factor(hopkins4)
table(hopkins4.f)
joincount.multi(hopkins4.f, nb2listw(hopkins.rook.nb, style="B"))
cat("replicates Upton & Fingleton table 3.4 (p. 166)\n")
joincount.multi(hopkins4.f, nb2listw(hopkins.queen.nb, style="B"))
cat("replicates Upton & Fingleton table 3.6 (p. 166)\n")
joincount.multi(hopkins4.f, nb2listw(hopkins.bishop.nb, style="B"))
cat("replicates Upton & Fingleton table 3.7 (p. 169)\n")

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

Value
A list with class jclist of lists with class htest for each of the k colours containing the following components:
statistic the value of the standard deviate of the join count statistic.
p.value the p-value of the test.
estimate the value of the observed statistic, its expectation and variance under non-free sampling.
alternative a character string describing the alternative hypothesis.
method a character string giving the method used.
data.name a character string giving the name(s) of the data.

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

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
joincount.mc, joincount.multi, listw2U
Examples

data(oldcol)
  HICRIME <- cut(COL.OLD$CRIME, breaks=c(0,35,80), labels=c("low","high"))
  names(HICRIME) <- rownames(COL.OLD)
  joincount.test(HICRIME, nb2listw(COL.nb, style="B"))
  joincount.test(HICRIME, nb2listw(COL.nb, style="B"), sampling="free")
  joincount.test(HICRIME, nb2listw(COL.nb, style="C"))
  joincount.test(HICRIME, nb2listw(COL.nb, style="S"))
  joincount.test(HICRIME, nb2listw(COL.nb, style="W"))
  by(card(COL.nb), HICRIME, summary)
  print(is.symmetric.nb(COL.nb))
  coords.OLD <- cbind(COL.OLD$X, COL.OLD$Y)
  COL.k4.nb <- knn2nb(knearneigh(coords.OLD, 4))
  print(is.symmetric.nb(COL.k4.nb))
  joincount.test(HICRIME, nb2listw(COL.k4.nb, style="B"))
  cat("Note non-symmetric weights matrix - use listw2U()
  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 points belonging to the set of the k nearest neighbours of each other. If longlat = TRUE, Great Circle distances are used. A warning will be given if identical points are found.

Usage

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

Arguments

x
  matrix of point coordinates, an object inheriting from SpatialPoints or an "sf" or "sfc" object; if the "sf" or "sfc" object geometries are in geographical coordinates (sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE), s2 will be used to find the neighbours because it uses spatial indexing https://github.com/r-spatial/spdep/issues/38 as opposed to the legacy method which uses brute-force

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; longlat will override kd_tree

use_kd_tree
  logical value, if the dbscan package is available, use for finding k nearest neighbours when longlat is FALSE, and when there are no identical points; from https://github.com/r-spatial/spdep/issues/38, the input data may have more than two columns if dbscan is used
Details

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

Value

A list of class knn

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

- knn, dnearneigh, knn2nb, kNN

Examples

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

xy1 <- SpatialPoints((as.data.frame(state.center))[m50.48,],
  proj4string=CRS("+proj=longlat +ellps=GRS80"))
gck4a.nb <- knn2nb(knearneigh(xy1, k=4))
summary(gck4a.nb, xy1, scale=0.5)
xy1 <- st_as_sf((as.data.frame(state.center))[m50.48,, coors=1:2,
crs=st_crs("+proj=longlat +ellps=GRS80")] old_use_s2 <- sf_use_s2()
sf_use_s2(TRUE)
system.time(gck4a.nb <- knn2nb(knearneigh(xy1, k=4)))
summary(gck4a.nb, xy1, scale=0.5)
sf_use_s2(FALSE)
system.time(gck4a.nb <- knn2nb(knearneigh(xy1, k=4)))
summary(gck4a.nb, xy1, scale=0.5)
sf_use_s2(old_use_s2)

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

---

knn2nb

## Neighbours list from knn object

### Description

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

### Usage

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

### Arguments

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

### Value

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

### Author(s)

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

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

#### Arguments

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

#### Value

a numeric vector the same length as var

#### Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

#### See Also

- `nb2listw`
Examples

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

---

### lee

**Compute Lee's statistic**

#### Description

A simple function to compute Lee’s L statistic for bivariate spatial data:

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

#### Usage

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

#### Arguments

- **x**: a numeric vector the same length as the neighbours list in listw
- **y**: a numeric vector the same length as the neighbours list in listw
- **listw**: a `listw` object created for example by `nb2listw`
- **n**: number of zones
- **S2**: Sum of squared sum of weights by rows.
- **zero.policy**: default `NULL`, use global option value; if `TRUE` assign zero to the lagged value of zones without neighbours, if `FALSE` assign `NA`
- **NAOK**: if `"TRUE"` then any `"NA"` or `"NaN"` or `"Inf"` values in `x` are passed on to the foreign function. If `"FALSE"`, the presence of `"NA"` or `"NaN"` or `"Inf"` values is regarded as an error.

#### Value

- a list of
  - `L`: Lee’s L statistic
  - `local L`: Lee’s local L statistic

#### Author(s)

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

See Also

lee.mc

Examples

```r
data(boston, package="spData")
lw<-nb2listw(boston.soI)
x<-boston.c$CMEDV
y<-boston.c$CRIM
z<-boston.c$RAD

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

Description

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

Usage

```r
lee.mc(x, y, 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
- **y**: a numeric vector the same length as the neighbours list in listw
- **listw**: a listw object created for example by nb2listw
- **nsim**: number of permutations
- **zero.policy**: default 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 Lee’s L.
- parameter  the rank of the observed Lee’s L.
- p.value  the pseudo p-value of the test.
- alternative  a character string describing the alternative hypothesis.
- method  a character string giving the method used.
- data.name  a character string giving the name(s) of the data, and the number of simulations.
- res  nsim simulated values of statistic, final value is observed statistic

Author(s)

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

References


See Also

lee

Examples

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

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

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

#Test with missing values
x[1:5]<-NA
Lee's L test for spatial autocorrelation using a spatial weights matrix in weights list form. The assumptions underlying the test are sensitive to the form of the graph of neighbour relationships and other factors, and results may be checked against those of `lee.mc` permutations.

### Usage

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

### Arguments

- **x**: a numeric vector the same length as the neighbours list in `listw`
- **y**: a numeric vector the same length as the neighbours list in `listw`
- **listw**: a `listw` object created for example by `nb2listw`
- **zero.policy**: default `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.
- **na.action**: a function (default `na.fail`), can also be `na.omit` or `na.exclude` - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set `zero.policy` to `TRUE` because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted. If `na.pass` is used, zero is substituted for NA values in calculating the spatial lag
- **spChk**: should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`

### Value

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

- **statistic**: the value of the standard deviate of Lee’s L.
- **p.value**: the p-value of the test.
- **estimate**: the value of the observed Lee’s L, its expectation and variance under the method assumption.
alternative  a character string describing the alternative hypothesis.
method      a character string giving the assumption used for calculating the standard devi- 
            ate.
data.name    a character string giving the name(s) of the data.

Note
See Lee (2004) for details on how the asymptotic expectation and variance of Lee’s L is computed.
In particular, check Lee (2004), table 1, page 1690.
This test may fail for large datasets as the computation of the asymptotic expectation and variance 
requires the use of dense matrices.

Author(s)
Roger Bivand and Virgilio GÃ³mez-Rubio <Virgilio.Gomez@uclm.es>

References
Lee (2004). A generalized significance testing method for global measures of spatial association: 

See Also
lee, lee.mc, listw2U

Examples

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

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

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

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

data(boston, package="spData")
lw<-nb2listw(boston.soi)
x<-boston.c$CMEDV
y<-boston.c$CRIM

lee.test(x, y, lw, zero.policy=TRUE, alternative="less")
#Test with missing values
x[1:5]<-NA
y[3:7]<-NA

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

## listw2sn

**Spatial neighbour sparse representation**

### Description

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

### Usage

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

### 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 coLid)
- `weights`: weight for this link
Lagrange Multiplier diagnostics for spatial dependence in linear models

Description

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

Usage

lm.LMtests(model, listw, zero.policy=NULL, test="LMerr", spChk=NULL, naSubset=TRUE)
## S3 method for class 'LMtestlist'
print(x, ...)
## S3 method for class 'LMtestlist'
print(x, digits=max(3, getOption("digits") - 2), ...)
Arguments

- **model**: an object of class `lm` returned by `lm`, or optionally a vector of externally calculated residuals (run through `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, RLM-lag, SARMA; test="all" computes all the tests.

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

- **naSubset**: default TRUE to subset listw object for omitted observations in model object (this is a change from earlier behaviour, when the `model$na.action` component was ignored, and the listw object had to be subsetted by hand).

- **x, object**: object to be printed.

- **p.adjust.method**: a character string specifying the probability value adjustment (see `p.adjust`) for multiple tests, default "none".

- **digits**: minimum number of significant digits to be used for most numbers.

- **...**: printing arguments to be passed through.

Details

The two types of dependence are for spatial lag $\rho$ and spatial error $\lambda$:

\[ y = X\beta + \rho W(1)y + u, \]
\[ u = \lambda W(2)u + e \]

where $e$ 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 $u$ test whether $\lambda = 0$. $W$ 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**


**See Also**

`lm`

**Examples**

data(oldcol)
oldcrime.lm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD)
summary(oldcrime.lm)
res <- lm.LMtests(oldcrime.lm, nb2listw(COL.nb), test=c("LMerr", "LMlag", "RLMerr", "RLMlag", "SARMA"))
summary(res)
res.LMtests(oldcrime.lm, nb2listw(COL.nb))
res.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 $\frac{1}{2}(W + W')$

**Usage**

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

**Arguments**

- **model**: an object of class `lm` returned by `lm`; weights may be specified in the `lm` fit, but offsets should not be used
- **listw**: a `listw` object created for example by `nb2listw`
- **zero.policy**: default 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.

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")),
       randomisation=FALSE)
oldcrime.wlm <- lm(CRIME ~ HOVAL + INC, data = COL.OLD,
       weights = I(1/AREA_PL))
lm.morantest(oldcrime.wlm, nb2listw(COL.nb, style="W"),
       resfun=weighted.residuals)
lm.morantest(oldercrime.wlm, nb2listw(COL.nb, style="W"),
resfun=rstudent)

## 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

Description

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

Usage

lm.morantest.exact(model, listw, zero.policy = NULL, alternative = "greater",
spChk = NULL, resfun = weighted.residuals, zero.tol = 1e-07, Omega=NULL,
save.M=NULL, save.U=NULL, useTP=FALSE, truncErr=1e-6, zeroTreat=0.1)
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

References


See Also

- `lm.morantest.sad`

Examples

```r
eire <- st_read(system.file("shapes/eire.shp", package="spData")[[1]])
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
eire.nb <- poly2nb(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
lm.morantest.exact(e.lm, nb2listw(eire.nb))
lm.morantest.exact(e.lm, nb2listw(eire.nb), useTP=TRUE)
```
lm.morantest.sad

Saddlepoint approximation of global Moran’s I test

Description

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

Usage

```r
lm.morantest.sad(model, listw, zero.policy=NULL, alternative="greater",
spChk=NULL, resfun=weighted.residuals, tol=.Machine$double.eps^0.5,
maxiter=1000, tol.bounds=0.0001, zero.tol = 1e-07, Omega=NULL,
save.M=NULL, save.U=NULL)
```

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

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

## S3 method for class summary.moransad
print(x, ...)

Arguments

- `model`: an object of class `lm` returned by `lm`; weights may be specified in the `lm` fit, but offsets should not be used.
- `listw`: a `listw` object created for example by `nb2listw`.
- `zero.policy`: default `NULL`, use global option value; if `TRUE` assign zero to the lagged value of zones without neighbours, if `FALSE` assign NA.
- `alternative`: a character string specifying the alternative hypothesis, must be one of greater (default), less or two.sided.
- `spChk`: should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default `NULL` to use `get.spChkOption()`.
- `resfun`: default: `weighted.residuals`; the function to be used to extract residuals from the `lm` object, may be `residuals`, `weighted.residuals`, `rstandard`, or `rstudent`.
- `tol`: the desired accuracy (convergence tolerance) for `uniroot`.
- `maxiter`: the maximum number of iterations for `uniroot`.
- `tol.bounds`: offset from bounds for `uniroot`.
- `zero.tol`: tolerance used to find eigenvalues close to absolute zero.
- `Omega`: A SAR process matrix may be passed in to test an alternative hypothesis, for example `Omega <- invIrW(listw, rho=0.1); Omega <- tcrossprod(Omega).chol()` is taken internally.
- `save.M`: return the full M matrix for use in `spdep::exactMoranAlt`.
- `save.U`: return the full U matrix for use in `spdep::exactMoranAlt`. 
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 &lt;Roger.Bivand@nhh.no&gt;

**References**


**See Also**

`lm.morantest`

**Examples**

eire <- st_read(system.file("shapes/eire.shp", package="spData")[[1]])
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
eire.nb <- poly2nb(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
lm.morantest(e.lm, nb2listw(eire.nb))
lm.morantest.sad(e.lm, nb2listw(eire.nb))
summary(lm.morantest.sad(e.lm, nb2listw(eire.nb)))
e.wlm <- lm(OWNCONS ~ ROADACC, data=eire, weights=RETSALE)
lm.morantest(e.wlm, nb2listw(eire.nb), resfun=rstudent)
lm.morantest.sad(e.wlm, nb2listw(eire.nb), resfun=rstudent)

---

localC

Compute Local Geary statistic

Description

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

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

Usage

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

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

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

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

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

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

localC_perm(x, ..., conditional=FALSE, zero.policy=NULL)

## Default S3 method:
localC_perm(x, listw, nsim = 499, alternative = "less", ..., conditional=FALSE, zero.policy=NULL)

## S3 method for class 'formula'
localC_perm(formula, listw, data, nsim = 499, alternative = "less", ..., conditional=FALSE, zero.policy=NULL)

Arguments

x a numeric vector, numeric matrix, or list. See details for more.

formula A one-sided formula determining which variables to be used.

listw a listw object created for example by nb2listw.

data Used when a formula is provided. A matrix or data frame containing the variables in the formula formula.

nsim The number of simulations to be used for permutation test.

alternative A character defining the alternative hypothesis. Must be one of "less" or "greater".

... other arguments passed to methods.

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

conditional default FALSE, sample from the whole data set; if TRUE, sample from all observations but the one which is being tested.

Details

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

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

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

Author(s)

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

References


Examples

```
orig <- spData::africa.rook.nb
listw <- nb2listw(orig)
x <- spData::afcon$totcon
(A <- localC(x, listw))
listw1 <- nb2listw(droplinks(sym.attr.nb(orig), 3, sym=TRUE), zero.policy=TRUE)
(A1 <- localC(x, listw1, zero.policy=FALSE))
(A2 <- localC(x, listw1, zero.policy=TRUE))
run <- FALSE
if (require(rgeoda, quietly=TRUE)) run <- TRUE
if (run) {
  W <- create_weights(as.numeric(length(x)))
  for (i in 1:length(listw$neighbours)) {
    set_neighbors_with_weights(W, i, listw$neighbours[[i]], listw$weights[[i]])
    update_weights(W)
  }
  set.seed(1)
  B <- local_geary(W, data.frame(x))
  all.equal(A, lisa_values(B))
}
if (run) {
  set.seed(1)
  C <- localC_perm(x, listw, nsim = 499, conditional=TRUE,
                   alternative="two.sided")
  cor(ifelse(lisa_pvalues(B) < 0.5, lisa_pvalues(B), 1-lisa_pvalues(B)),
      attr(C, "pseudo-p")[,6])
}
```

# pseudo-p values probably wrongly folded https://github.com/GeoDaCenter/rgeoda/issues/28
## Not run:
library(reticulate)
use_python("/usr/bin/python", required = TRUE)
gp <- import("geopandas")
ps <- import("libpysal")
tf <- tempfile(fileext=".gal")
write.nb.gal(orig, tf)
fl <- ps$io$open(tf)
w$transform <- "R"
esda <- import("esda")
lM <- esda$Moran_Local(x, w)
all.equal(unname(localmoran(x, listw, mlvar=FALSE)[,1]), c(lM$Is))
# confirm x and w the same
lC <- esda$Geary_Local(connectivity=w)$fit(scale(x))
# np$std missing ddof=1
n <- length(x)
D01 <- spdep:::geary.intern((x - mean(x)) / sqrt(var(x)*(n-1)/n), listw, n=n)
# lC components probably wrongly ordered https://github.com/pysal/esda/issues/192
o <- match(round(D0, 6), round(lC$localG, 6))
all.equal(c(lC$localG)[o], D0)
# simulation order not retained
lC$p_sim[o]
attr(C, "pseudo-p")[,6]
```
localG

## End(Not run)

### localG

---

**Description**

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

**Usage**

```r
localG(x, listw, zero.policy=NULL, spChk=NULL, return_internals=FALSE, GeoDa=FALSE)
localG_perm(x, listw, nsim=499, zero.policy=NULL, spChk=NULL, return_internals=FALSE, iseed=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()`
- `return_internals`: default `FALSE`, if `TRUE`, return internal values of $G$, EI and VG as as attribute matrix
- `GeoDa`: default `FALSE`, if `TRUE`, drop x values for no-neighbour and self-neighbour only observations from all summations
- `nsim`: default `499`, number of conditional permutation simulations
- `iseed`: default `NULL`, used to set the seed for possible parallel RNGs

**Details**

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

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

A vector of G or Gstar values, with attributes "gstari" set to TRUE or FALSE, "call" set to the function call, and class "localG".

Note

Conditional permutations added for comparative purposes; permutations are over the whole data vector omitting the observation itself.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


Examples

data(getisord, package="spData")
# spData 0.3.2 changes x, y, xyz object names to go_x, go_y, go_xyz to
# avoid putting these objects into the global environment via lazy loading
if (exists("go_xyz") && packageVersion("spData") >= "0.3.2") {
    xyz <- go_xyz
    x <- go_x
    y <- go_y
}
xycoords <- cbind(xyz$x, xyz$y)
nb30 <- dnearest(xycoords, 0, 30)
G30 <- localG(xyz$val, nb2listw(nb30, style="B"))
G30[length(xyz$val)-136]
set.seed(1)
G30_sim <- localG_perm(xyz$val, nb2listw(nb30, style="B"))
G30_sim[length(xyz$val)-136]
nb60 <- dnearest(xycoords, 0, 60)
G60 <- localG(xyz$val, nb2listw(nb60, style="B"))
G60[length(xyz$val)-136]
nb90 <- dnearest(xycoords, 0, 90)
G90 <- localG(xyz$val, nb2listw(nb90, style="B"))
G90[length(xyz$val)-136]
nb120 <- dnearest(xycoords, 0, 120)
G120 <- localG(xyz$val, nb2listw(nb120, style="B"))
G120[length(xyz$val)-136]
nb150 <- dnearest(xycoords, 0, 150)
G150 <- localG(xyz$val, nb2listw(nb150, style="B"))
G150[length(xyz$val)-136]
localmoran <- seq(-5, 5, 1)
cm.col <- cm.colors(length(localmoran)-1)
image(x, y, t(matrix(G30, nrow=16, ncol=16, byrow=TRUE)),
      breaks=localmoran, 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=localmoran, 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:

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

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

Usage

localmoran(x, listw, zero.policy=NULL, na.action=na.fail, conditional=TRUE,
            alternative = "two.sided", p.adjust.method="none", mlvar=TRUE,
            spChk=NULL, adjust.x=FALSE)
localmoran_perm(x, listw, nsim=499, zero.policy=NULL, na.action=na.fail,
                alternative = "two.sided", p.adjust.method="none", mlvar=TRUE,
                spChk=NULL, adjust.x=FALSE, sample_Ei=TRUE, iseed=NULL)

Arguments

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

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.*)

conditional default TRUE: expectation and variance are calculated using the conditional randomization null (Sokal 1998 Eqs. A7 & A8). Elaboration of these changes available in Sauer et al. (2021). If FALSE: expectation and variance are calculated using the total randomization null (Sokal 1998 Eqs. A3 & A4).

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

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()

adjust.x default FALSE, if TRUE, x values of observations with no neighbours are omitted in the mean of x

nsim default 499, number of conditonal permutation simulations

sample_Ei default TRUE; if conditional permutation, use the sample $E_i$ values, or the analytical values, leaving only variances calculated by simulation.

iseed default NULL, used to set the seed for possible parallel RNGs

Details

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

Value

Ii local moran statistic
**localmoran**

**E.Ii** expectation of local moran statistic; for localmoran_perm the permutation sample means

**Var.Ii** variance of local moran statistic; for localmoran_perm the permutation sample standard deviations

**Z.Ii** standard deviate of local moran statistic; for localmoran_perm based on permutation sample means and standard deviations

**Pr()** p-value of local moran statistic using pnorm(); for localmoran_perm using standard deviation based on permutation sample means and standard deviations

**Pr() Sim** For localmoran_perm, rank() and punif() of observed statistic rank for [0, 1] p-values using alternative=

**Pr(folded) Sim** the simulation folded [0, 0.5] range ranked p-value (based on https://github.com/pysal/esda/blob/4a63e0b5df1e754b17b5f1205b8cadcbec5e061/esda/crand.py#L211-L213)

**Skewness** For localmoran_perm, the output of e1071::skewness() for the permutation samples underlying the standard deviates

**Kurtosis** For localmoran_perm, the output of e1071::kurtosis() for the permutation samples underlying the standard deviates

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

**Note**

Conditional permutations added for comparative purposes; permutations are over the whole data vector omitting the observation itself.

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**References**


**See Also**

localG
Examples

data(afcon, package="spData")
oid <- order(afcon$id)
resI <- localmoran(afcon$totcon, nb2listw(paper.nb))

printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]),
  check.names=FALSE)
hist(resI[,5])

mean(resI[,1])

sum(resI[,1])/Szero(nb2listw(paper.nb))

moran.test(afcon$totcon, nb2listw(paper.nb))

# note equality for mean() only when the sum of weights equals
# the number of observations (thanks to Juergen Symanzik)

resI <- localmoran(afcon$totcon, nb2listw(paper.nb),
  p.adjust.method="bonferroni")

printCoefmat(data.frame(resI[oid,], row.names=afcon$name[oid]),
  check.names=FALSE)
hist(resI[,5])

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

totcon

resI.na <- localmoran(totcon, nb2listw(paper.nb), na.action=na.exclude,
  zero.policy=TRUE)

if (class(attr(resI.na, "na.action")) == "exclude") {
  print(data.frame(resI.na[oid,], row.names=afcon$name[oid]), digits=2)
} else print(resI.na, digits=2)

resG <- localG(afcon$totcon, nb2listw(include.self(paper.nb)))

print(data.frame(resG[oid], row.names=afcon$name[oid]), digits=2)

set.seed(1)

resI_p <- localmoran_perm(afcon$totcon, nb2listw(paper.nb))

printCoefmat(data.frame(resI_p[oid,], row.names=afcon$name[oid]),
  check.names=FALSE)

---

localmoran.exact

Exact local Moran's Ii tests

Description

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

Usage

localmoran.exact(model, select, nb, glist = NULL, style = "W",
  zero.policy = NULL, alternative = "greater", spChk = NULL,
  resfun = weighted.residuals, save.Vi = FALSE, useTP=FALSE, truncErr=1e-6,
  zeroTreat=0.1)

localmoran.exact.alt(model, select, nb, glist = NULL, style = "W",
  zero.policy = NULL, alternative = "greater", spChk = NULL,
  resfun = weighted.residuals, save.Vi = FALSE, useTP=FALSE, truncErr=1e-6,
  zeroTreat=0.1)
localmoran.exact

zero.policy = NULL, alternative = "greater", spChk = NULL, resfun = weighted.residuals, Omega = NULL, save.Vi = FALSE, save.M = FALSE, useTP=FALSE, truncErr=1e-6, zeroTreat=0.1)
## S3 method for class 'localmoranex'
print(x, ...)
## S3 method for class 'localmoranex'
as.data.frame(x, row.names=NULL, optional=FALSE, ...)

Arguments

model an object of class 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 `localmoran.exact` containing "select" lists, each with class `moran.exact` with the following components:

- **statistic**: the value of the exact standard deviate of global Moran’s I.
- **p.value**: the p-value of the test.
- **estimate**: the value of the observed local Moran’s 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

References


See Also

`lm.morantest.exact, localmoran.sad`

Examples

eire <- st_read(system.file("shapes/eire.shp", package="spData")[[1]])
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
eire.nb <- poly2nb(eire)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
localmoran.sad(e.lm, nb=eire.nb)
localmoran.exact(e.lm, nb=eire.nb)
localmoran.exact(e.lm, nb=eire.nb, useTP=TRUE)
run <- FALSE
if (requireNamespace("spatialreg", quietly=TRUE)) run <- TRUE
if (run) {

```
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)

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 \(I_i\).
- **p.value**: the p-value of the test.
- **estimate**: the value of the observed local Moran’s \(I_i\).
- **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
- **df**: degrees of freedom
- **tau**: maximum and minimum analytical eigenvalues
- **i**: zone tested

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

- `localmoran`
- `lm.morantest`
- `lm.morantest.sad`
- `errorsarlm`

Examples

eire <- st_read(system.file("shapes/eire.shp", package="spData")[[1]])
row.names(eire) <- as.character(eire$names)
st_crs(eire) <- "+proj=utm +zone=30 +ellps=airy +units=km"
eire.nb <- poly2nb(eire)
lw <- nb2listw(eire.nb)
e.lm <- lm(OWNCONS ~ ROADACC, data=eire)
e.locmor <- summary(localmoran.sad(e.lm, nb=eire.nb))
e.locmor
mean(e.locmor[,1])
sum(e.locmor[,1])/Szero(lw)
lm.morantest(e.lm, lw)
# note equality for mean() only when the sum of weights equals
# the number of observations (thanks to Juergen Symanzik)
hist(e.locmor[,"Pr. (Sad)"])
e.wlm <- lm(OWNCONS ~ ROADACC, data=eire, weights=RETSALE)
e.locmorw1 <- summary(localmoran.sad(e.wlm, nb=eire.nb, resfun=weighted.residuals))
e.locmorw1
Local spatial heteroscedasticity is calculated for each location based on the spatial weights object used. The statistic is:

\[ H_i = \frac{\sum_j w_{ij} \cdot |e_j|^a}{\sum_j w_{ij}} \]

with

\[ e_j = x_j - \bar{x}_j \]

and

\[ \bar{x}_j = \frac{\sum_k w_{jk} \cdot x_k}{\sum_k w_{jk}} \]

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

Usage

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

**Arguments**

- **x**: a numeric vector of the same length as the neighbours list in listw
- **listw**: a listw object created for example by `nb2listw`
- **a**: the exponent applied to the local residuals; the default value of 2 leads to a measure of heterogeneity in the spatial variance
- **var_hi**: default TRUE, the moments and the test statistics are calculated for each location; if FALSE, only the plain LOSH measures, \( \bar{x}_i \) and \( e_i \) are calculated
- **zero.policy**: default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
- **na.action**: a function (default `na.fail`), can also be `na.omit` or `na.exclude` - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted. If `na.pass` is used, zero is substituted for NA values in calculating the spatial lag. (Note that `na.exclude` will only work properly starting from R 1.9.0, `na.omit` and `na.exclude` assign the wrong classes in 1.8.*)
- **spChk**: should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`

**Details**

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

**Value**

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

**Author(s)**

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

**References**

**LOSH.cs**

Chi-square based test for local spatial heteroscedasticity

**Description**

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

**Usage**

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

**Arguments**

- `x` a numeric vector of the same length as the neighbours list in `listw`
- `listw` a `listw` object created for example by `nb2listw`
- `zero.policy` default `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.*)
- `p.adjust.method` a character string specifying the probability value adjustment for multiple tests, default "none"; see `p.adjustSP`. Note that the number of multiple tests for each region is only taken as the number of neighbours + 1 for each region, rather than the total number of regions.
- `spChk` should the data vector names be checked against the spatial objects for identity integrity, `TRUE`, or `FALSE`, default `NULL` to use `get.spChkOption()`

**Examples**

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

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

Value

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

Author(s)

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

References


See Also

\texttt{LOSH, LOSH.mc}

Examples

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

---

**LOSH.mc**

*Bootstrapping-based test for local spatial heteroscedasticity*

Description

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

Usage

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

- **x**: a numeric vector of the same length as the neighbours list in listw
- **listw**: a listw object created for example by nb2listw
- **a**: the exponent applied to the local residuals; the default value of 2 leads to a measure of heterogeneity in the spatial variance
- **nsim**: the number of randomisations used in the bootstrap
- **zero.policy**: default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
- **na.action**: a function (default na.fail), can also be na.omit or na.exclude - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted. If na.pass is used, zero is substituted for NA values in calculating the spatial lag. (Note that na.exclude will only work properly starting from R 1.9.0, na.omit and na.exclude assign the wrong classes in 1.8.*)
- **spChk**: should the data vector names be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
- **adjust.n**: default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted
- **p.adjust.method**: a character string specifying the probability value adjustment for multiple tests, default "none"; see p.adjustSP. Note that the number of multiple tests for each region is only taken as the number of neighbours + 1 for each region, rather than the total number of regions.

Details

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

Value

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

Author(s)

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


See Also

LOSH, LOSH.mc

Examples

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

mat2listw

Convert a square spatial weights matrix to a weights list object

Description

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

Compute Moran’s I

Description

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

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

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
moran.mc

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


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))
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`
- **adjust.n**: default TRUE, if FALSE the number of observations is not adjusted for no-neighbour observations, if TRUE, the number of observations is adjusted

Value

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

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

`moran`, `moran.test`
Examples

data(oldcol)
colw <- nb2listw(COL.nb, style="W")
nsim <- 99
set.seed(1234)
sim1 <- moran.mc(COL.OLD$CRIME, listw=colw, nsim=nsim)
sim1
mean(sim1$res[1:nsim])
var(sim1$res[1:nsim])
summary(sim1$res[1:nsim])
colold.lags <- nblag(COL.nb, 3)
set.seed(1234)
sim2 <- moran.mc(COL.OLD$CRIME, nb2listw(colold.lags[[2]],
    style="W"), nsim=nsim)
summary(sim2$res[1:nsim])
sim3 <- moran.mc(COL.OLD$CRIME, nb2listw(colold.lags[[3]],
    style="W"), nsim=nsim)
summary(sim3$res[1:nsim])

moran.plot
Moran scatterplot

Description

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

Usage

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

Arguments

x a numeric vector the same length as the neighbours list in listw
listw a listw object created for example by nb2listw
zero.policy default NULL, use global option value; if TRUE assign zero to the lagged value
of zones without neighbours, if FALSE assign NA
spChk should the data vector names be checked against the spatial objects for identity
integrity, TRUE, or FALSE, default NULL to use get.spChkOption()
labels character labels for points with high influence measures, if set to FALSE, no
labels are plotted for points with large influence
xlab label for x axis
ylab label for x axis
quiet  default NULL, use !verbose global option value; if TRUE, output of summary of influence object suppressed
plot   default TRUE, if false, plotting is suppressed
return_df default TRUE, invisibly return a data.frame object; if FALSE invisibly return an influence measures object
... further graphical parameters as in par(..)

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

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
localmoran, influence.measures

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

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, drop.EI2=FALSE)

Arguments

x a numeric vector the same length as the neighbours list in listw
listw a listw object created for example by nb2listw
randomisation variance of I calculated under the assumption of randomisation, if FALSE normality
zero.policy default 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
drop.EI2 default FALSE, if TRUE, emulate CrimeStat <= 4.02

Value

A list with class htest containing the following components:

statistic the value of the standard deviate of Moran’s I.
moran.test

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 devi-
ate.

data.name a character string giving the name(s) of the data.

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

moran, moran.mc, listw2U

Examples

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

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

mstree

Find the minimal spanning tree

Description

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

Usage

mstree(nbw, ini = NULL)

Arguments

nbw  An object of listw class returned by 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


Examples

```r
### loading data
bh <- st_read(system.file("etc/shapes/bhicv.shp", package="spdep")[[1]], quiet=TRUE)
st_crs(bh) <- "+proj=longlat +ellps=WGS84"
### data padronized
dpad <- data.frame(scale(as.data.frame(bh)[,5:8]))
### neighboorhod list
bh.nb <- poly2nb(bh)
### calculing costs
lcosts <- nbcosts(bh.nb, dpad)
### making listw
nb.w <- nb2listw(bh.nb, lcosts, style="B")
### find a minimum spanning tree
system.time(mst.bh <- mstree(nb.w,5))
dim(mst.bh)
head(mst.bh)
tail(mst.bh)
### the mstree plot
par(mar=c(0,0,0,0))
plot(st_geometry(bh), border=gray(.5))
plot(mst.bh, coordinates(as(bh, "Spatial")), col=2,
     cex.lab=.6, cex.circles=0.035, fg="blue", add=TRUE)
```
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 functions

nb.obj1

A neighbor list created from any of the neighborhood list functions

nb.obj2

A neighbor list created from any of the neighborhood list functions

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

A new neighborlist created from the set operations on the input neighbor list(s)

Author(s)

Nicholas Lewin-Koh <nikko@hailmail.net>

See Also

intersect.nb, union.nb, setdiff.nb

Examples

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
coords <- coordinates(as(columbus, "Spatial"))
col.tri.nb <- tri2nb(coords)
oldpar <- par(mfrow=c(1,2))
if (require("dbscan", quietly=TRUE)) {
col.soi.nb <- graph2nb(soi.graph(col.tri.nb, coords))
plot(st_geometry(columbus), border="grey")
plot(col.soi.nb, coords, add=TRUE)
nb2blocknb

Block up neighbour list for location-less observations

Description

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

Usage

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

Arguments

nb an object of class nb with a list of integer vectors containing neighbour region number ids; if null, an nb object with no neighbours is created the length of unique(as.character(ID))

ID identification tags of the locations for the location-less observations; sort(unique(as.character(ID))) must be identical to sort(as.character(attr(nb,"region.id"))); same length as row.names if provided.

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

Details

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

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

knn2nb, dnearneigh, cell2nb, tri2nb, poly2nb

Examples

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

## End(Not run)
```
nb2INLA  

Output spatial neighbours for INLA

Description

Output spatial neighbours for INLA

Usage

nb2INLA(file, nb)

Arguments

file  
file where adjacency matrix will be stored

nb  
an object of class nb

Value

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

Author(s)

Virgilio Gomez-Rubio

References

http://www.r-inla.org

Examples

col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
td <- tempdir()
x <- nb2INLA(paste(td, "columbus-INLA.adj", sep="/"), col.gal.nb)
readLines(paste(td, "columbus-INLA.adj", sep="/", n=10)
nb2lines

Use vector files for import and export of weights

Description

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

Usage

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

Arguments

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

Details

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

Value

nb2lines and listw2lines return a SpatialLinesDataFrame object or an sf object; the data frame
contains with the from and to indices of the neighbour links and their weights. df2sn converts the
data retrieved from reading the data from df back into a spatial.neighbour object.
nb2listw

Spatial weights for neighbours lists

Description

The nb2listw function supplements a neighbours list with spatial weights for the chosen coding scheme. The can.be.simmed helper function checks whether a spatial weights object is similar to symmetric and can be so transformed to yield real eigenvalues or for Cholesky decomposition.

Usage

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

Arguments

neighbours an object of class nb

style can take values “W”, “B”, “C”, “U”, “minmax” and “S”

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

columns <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
res <- listw2lines(nb2listw(col.gal.nb), st_geometry(columbus))
summary(res)
tf <- paste0(tempfile(), ".gpkg")
st_write(res, dsn=tf, driver="GPKG")
inMap <- st_read(tf)
summary(inMap)
diffnb(sn2listw(df2sn(as.data.frame(inMap)))$neighbours, col.gal.nb)
res1 <- listw2lines(nb2listw(col.gal.nb), as(columbus, "Spatial"))
summary(res1)
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 \( t(rep(0, length=length(neighbours))) \times x \), for arbitrary numerical vector \( x \) of length \( length(neighbours) \). The spatially lagged value of \( x \) for the zero-neighbour region will then be zero, which may (or may not) be a sensible choice.

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

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

Value

A listw object with the following members:

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

summary.nb, read.gal
Examples

```r
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
coords <- coordinates(as(columbus, "Spatial"))
cards <- card(col.gal.nb)
col.w <- nb2listw(col.gal.nb)
plot(cards, unlist(lapply(col.w$weights, sum)), xlim=c(0,10),
     ylim=c(0,10), xlab="number of links", ylab="row sums of weights")
col.w <- nb2listw(col.gal.nb, style="B")
points(cards, unlist(lapply(col.w$weights, sum)), col="red")
col.c <- nb2listw(col.gal.nb, style="C")
points(cards, unlist(lapply(col.c$weights, sum)), col="green")
col.u <- nb2listw(col.gal.nb, style="U")
points(cards, unlist(lapply(col.u$weights, sum)), col="orange")
col.s <- nb2listw(col.gal.nb, style="S")
points(cards, unlist(lapply(col.s$weights, sum)), col="blue")
legend(x=c(0, 1), y=c(7, 9), legend=c("W", "B", "C", "U", "S"), bty="n",
     col=c("black", "red", "green", "orange", "blue"), pch=rep(1,5), cex=0.8,
     y.intersp=2.5)
summary(nb2listw(col.gal.nb, style="minmax"))
dlist <- ndists(col.gal.nb, coords)
dlist <- lapply(dlist, function(x) 1/x)
col.w.d <- nb2listw(col.gal.nb, glist=dlist)
summary(unlist(col.w$weights))
summary(unlist(col.w.d$weights))
# introducing other conditions into weights - only earlier sales count
# see http://sal.uiuc.edu/pipermail/openspace/2005-October/000610.html
data(baltimore, package="spData")
set.seed(211)
dates <- sample(1:500, nrow(baltimore), replace=TRUE)
b_15nn <- knn2nb(knearneigh(cbind(baltimore$X, baltimore$Y), k=15))
glist <- vector(mode="list", length=length(b_15nn))
for (i in seq(along=b_15nn))
  glist[[i]] <- ifelse(dates[i] > dates[b_15nn[[i]]], 1, 0)
listw_15nn_dates <- nb2listw(b_15nn, glist=glist, style="B")
which(lag(listw_15nn_dates, baltimore$PRICE) == 0.0)
which(sapply(glist, sum) == 0)
ex <- which(sapply(glist, sum) == 0)[1]
dates[ex]
dates[b_15nn[[ex]]]
```

---

**nb2listwdist**  
*Distance-based spatial weights for neighbours lists*

**Description**

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

Usage

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

Arguments

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

Details

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

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

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

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

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

Value

A listw object with the following members:

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

Author(s)

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

References


See Also

nb2listw, summary.nb

Examples

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

### 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**

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

**Arguments**

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

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

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

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

nb2listw

Examples

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

nb2WB

Output spatial weights for WinBUGS

Description

Output spatial weights for WinBUGS

Usage

nb2WB(nb)
listw2WB(listw)
Compute cost of edges

Description

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

Usage

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

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

Value

A object of nbdist class. See nbdist for details.

Note

The neighbours must be a connected graph.

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as nbdist, nb2listw

---

nbdists  

*Spatial link distance measures*

Description

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

Usage

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

Arguments

nb

an object of class nb

coords

matrix of point coordinates, an object inheriting from SpatialPoints or an "sf" or "sfc" object; if the "sf" or "sfc" object geometries are in geographical coordinates (sf::st_is_longlat(x) == TRUE and sf::sf_use_s2() == TRUE), s2 will be used to find distances https://github.com/r-spatial/s2/issues/125

longlat

TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if coords is a SpatialPoints object, the value is taken from the object itself

Value

A list with class nbdist

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

summary.nb, nb2listw

Examples

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

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

```r
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
coords <- coordinates(as(columbus, "Spatial"))
summary(col.gal.nb, coords)
col.lags <- nblag(col.gal.nb, 2)
print(col.lags)
summary(col.lags[[2]], coords)
plot(st_geometry(columbus), border="grey")
plot(col.gal.nb, coords, add=TRUE)
title(main="GAL order 1 (black) and 2 (red) links")
plot(col.lags[[2]], coords, add=TRUE, col="red", lty=2)
cuml <- nblag_cumul(col.lags)
cuml
run <- FALSE
if (require(igraph, quietly=TRUE)) run <- TRUE
if (run) {
  W <- as(nb2listw(col.gal.nb), "CsparseMatrix")
  G <- graph.adjacency(W, mode="directed", weight="W")
  D <- diameter(G)
  nbs <- nblag(col.gal.nb, maxlag=D)
  n <- length(col.gal.nb)
  lmat <- lapply(nbs, nb2mat, style="B", zero.policy=TRUE)
  mat <- matrix(0, n, n)
  for (i in seq(along=lmat)) mat = mat + i*lmat[[i]]
  G2 <- shortest.paths(G)
  print(all.equal(G2, mat, check.attributes=FALSE))
}
```
Description

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

Usage

data(oldcol)

Format

This data frame contains the following columns:

- **AREA_PL**: computed by ArcView (agrees with areas of polygons in the “columbus” data set)
- **PERIMETER**: computed by ArcView
- **COLUMBUS**: internal polygon ID (ignore)
- **COLUMBUS.I**: another internal polygon ID (ignore)
- **POLYID**: yet another polygon ID
- **NEIG**: neighborhood id value (1-49); conforms to id value used in Spatial Econometrics book.
- **HOVAL**: housing value (in \$1,000)
- **INC**: household income (in \$1,000)
- **CRIME**: residential burglaries and vehicle thefts per thousand households in the neighborhood
- **OPEN**: open space in neighborhood
- **PLUMB**: percentage housing units without plumbin
- **DISCBD**: distance to CBD
- **X**: x coordinate (in arbitrary digitizing units, not polygon coordinates)
- **Y**: y coordinate (in arbitrary digitizing units, not polygon coordinates)
- **AREA_SS**: neighborhood area (computed by SpaceStat)
- **NSA**: north-south dummy (North=1)
- **NSB**: north-south dummy (North=1)
- **EW**: east-west dummy (East=1)
- **CP**: core-periphery dummy (Core=1)
- **THOUS**: constant=1,000
- **NEIGNO**: NEIG+1,000, alternative neighborhood id value
- **PERIM**: polygon perimeter (computed by SpaceStat)
Details

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

Note

All source data files prepared by Luc Anselin, Spatial Analysis Laboratory, Department of Agricultural and Consumer Economics, University of Illinois, Urbana-Champaign, [https://spatial.uchicago.edu/sample-data](https://spatial.uchicago.edu/sample-data).

Source


---

**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

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

plot.mst

Plot the Minimum Spanning Tree

Description

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

Usage

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

Arguments

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

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as `skater` and `mstree`

Examples

```r
### 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**

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

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

**Arguments**

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

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**See Also**

`summary.nb`

**Examples**

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

---

plot.skater

*Plot the object of skater class*

**Description**

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

**Usage**

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

**Arguments**

- `x` An object of skater class.
- `coords` A matrix of two columns with coordinates of nodes.
- `label.areas` A vector of labels of nodes.
- `groups.colors` A vector with colors of groups or 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 functions.

**Author(s)**

Elias T. Krainski and Renato M. Assuncao

**See Also**

See Also as `skater` and `mstree`

**Examples**

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

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

Arguments

- `pl` list of polygons of class extending SpatialPolygons, or an sf or sfc object containing non-empty (multi-)polygon objects
- `row.names` character vector of region ids to be added to the neighbours list as attribute region.id, default seq(1,nrow(x)); if pl has row.names, they are used instead of the default sequence.
- `snap` boundary points less than snap distance apart are considered to indicate contiguity; used both to find candidate and actual neighbours for planar geometries, but only actual neighbours for spherical geometries, as spherical spatial indexing itself injects some fuzzyness.
- `queen` if TRUE, a single shared boundary point meets the contiguity condition, if FALSE, more than one shared point is required; note that more than one shared boundary point does not necessarily mean a shared boundary line
- `useC` default TRUE, doing the work loop in C, may be set to false to revert to R code calling two C functions in an n*k work loop, where k is the average number of candidate neighbours
- `foundInBox` default NULL using R code or st_intersects() to generate candidate neighbours (using snap= if the geometries are not spherical); if not NULL (for legacy purposes) a list of length (n-1) with integer vectors of candidate neighbours (j > i) (as created by the poly_findInBoxGEOS function in rgeos for clean polygons)

Value

A neighbours list with class nb. See `card` for details of “nb” objects.
Note

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

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

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

Author(s)

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

See Also

summary.nb, card

Examples

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
coords <- st_coordinates(st_centroid(st_geometry(columbus)))
xx <- poly2nb(as(columbus, "Spatial"))
dxx <- diffnb(xx, col.gal.nb)
plot(st_geometry(columbus), border="grey")
plot(col.gal.nb, coords, add=TRUE)
plot(dxx, coords, add=TRUE, col="red")
title(main=paste("Differences (red) in Columbus GAL weights (black)",
              "and polygon generated queen weights", sep="\n"), cex.main=0.6)

# poly2nb with sf sfc_MULTIPOLYGON objects
sf_xx <- poly2nb(columbus)
diffnb(sf_xx, xx)
sfc_xx <- poly2nb(st_geometry(columbus))
diffnb(sfc_xx, xx)

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

---

**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: probability of getting a more “extreme” count than actually observed - one-tailed, default alternative observed “less” than expected

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

EBest, EBlocal, ppois

Examples

auckland <- st_read(system.file("shapes/auckland.shp", package="spData")[[1]], quiet=TRUE)
res <- probmap(auckland$M77_85, 9*auckland$Und5_81)
rt <- sum(auckland$M77_85)/sum(9*auckland$Und5_81)
ppois_pmap <- numeric(length(auckland$Und5_81))
for (i in seq(along=ppois_pmap)) {
  ppois_pmap[i] <- poisson.test(auckland$M77_85[i], r=rt,
      T=(9*auckland$Und5_81)[i]), alternative="less")$p.value
  all.equal(ppois_pmap, res$pmap)
}
res$id <- 1:nrow(res)
auckland$id <- res$id <- 1:nrow(res)
auckland_res <- merge(auckland, res, by="id")
plot(auckland_res[, "raw"], main="Crude (raw) estimates")
plot(auckland_res[, "relRisk"], main="Standardised mortality ratios")
plot(auckland_res[, "pmap"], main="Poisson probabilities",
      breaks=c(0, 0.05, 0.1, 0.5, 0.9, 0.95, 1))

---

**prunecost**

*Compute cost of prune each edge*

Description

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

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

Arguments

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

Value

A 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)
Prune a Minimun Spanning Tree

Description

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

Usage

prunemst(edges, only.nodes = TRUE)

Arguments

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

Value

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

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as mstree

Examples

e <- matrix(c(2,3, 1,2, 3,4, 4,5), ncol=2, byrow=TRUE)
e
prunemst(e)
prunemst(e, only.nodes=FALSE)
read.gal Read a GAL lattice file into a neighbours list

Description

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

Usage

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

Arguments

file name of file with GAL lattice data
region.id region IDs in specified order to coarse 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


Value

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

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>
read.gwt2nb

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

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

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
read.gal

Examples
data(baltimore, package="spData")
STATION <- baltimore$STATION
gwt1 <- read.gwt2nb(system.file("weights/baltk4.GWT", package="spData")[1], STATION)
cat(paste("Neighbours list symmetry;", is.symmetric.nb(gwt1, FALSE, TRUE), "\n"))
listw1 <- nb2listw(gwt1, style="B", glist=attr(gwt1, "GeoDa")$dist)
tmpfile <- tempfile()
write.sn2gwt(listw2sn(listw1), tmpfile)
gwt2 <- read.gwt2nb(tmpfile, STATION)
cat(paste("Neighbours list symmetry;", is.symmetric.nb(gwt2, FALSE, TRUE), "\n"))
diffnb(gwt1, gwt2)
data(oldcol)
Rotation

Rotate a set of point by a certain angle

Description

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

Usage

Rotation(xy, angle)

Arguments

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

Value

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

Author(s)

F. Guillaume Blanchet

Examples

set.seed(1)
### Create a set of coordinates
coords <- cbind(runif(20), runif(20))
### Create a series of angles
rad <- seq(0, pi, l=20)

opar <- par(mfrow=c(5,4), mar=c(3,3,1,1))
for(i in rad){
  coords.rot <- Rotation(coords, i)
  plot(coords.rot, xlab="", ylab="")
}
par(opar)
### Rotate the coordinates by an angle of 90 degrees

```r
coords.90 <- Rotation(coords, 90*pi/180)
coords.90
plot(coords, xlim=range(rbind(coords.90,coords)[,1]),
ylim=range(rbind(coords.90,coords)[,2]), asp=1)
points(coords.90, pch=19)
```

---

**set.mcOption**

**Options for parallel support**

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

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

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

\begin{verbatim}
ls(envir=spdep:::.spdepOptions)
library(parallel)
nc <- detectCores(logical=FALSE)
nc
# set nc to 1L here
if (nc > 1L) nc <- 1L
#nc <- ifelse(nc > 2L, 2L, nc)
coresOpt <- get.coresOption()
coresOpt
if (!is.na(nc)) {
  invisible(set.coresOption(nc))
  print(exists("aple.mc"))
  if (.Platform$OS.type == "windows") {
    # forking not permitted on Windows - start cluster
    print(get.mcOption())
    cl <- makeCluster(get.coresOption())
    print(clusterEvalQ(cl, exists("aple.mc")))
    set.ClusterOption(cl)
    clusterEvalQ(get.ClusterOption(), library(spdep))
    print(clusterEvalQ(cl, exists("aple.mc")))
    clusterEvalQ(get.ClusterOption(), detach(package:spdep))
    set.ClusterOption(NULL)
    print(clusterEvalQ(cl, exists("aple.mc")))
    stopCluster(cl)
  } else {
    mcOpt <- get.mcOption()
    print(mcOpt)
    print(mclapply(1:get.coresOption(), function(i) exists("aple.mc"),
               mc.cores=get.coresOption()))
    invisible(set.mcOption(FALSE))
    cl <- makeCluster(nc)
    print(clusterEvalQ(cl, exists("aple.mc")))
    set.ClusterOption(cl)
    clusterEvalQ(get.ClusterOption(), library(spdep))
    print(clusterEvalQ(cl, exists("aple.mc")))
    clusterEvalQ(get.ClusterOption(), detach(package:spdep))
    set.ClusterOption(NULL)
    print(clusterEvalQ(cl, exists("aple.mc")))
    stopCluster(cl)
    invisible(set.mcOption(mcOpt))
  }
  invisible(set.coresOption(coresOpt))
\end{verbatim}
set.spChkOption

Control checking of spatial object IDs

Description

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

Usage

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

Arguments

check a logical value, TRUE or FALSE
x a vector the same length, or a two-dimensional array, or data frame with the same number of rows as the neighbours list in listw
listw a listw object or nb object inheriting from "nb"
var a character string or integer value for the column to be selected
data a two-dimensional array or data frame containing var

Details

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

Value

set.spChkOption() returns the old logical value, get.spChkOption() returns the current logical value, and chkIDs() returns a logical value for the test lack of difference. spNamedVec() returns the selected column with the names set to the row names of the object from which it has been extracted.
The motivation for this mechanism is provided by the observation that spatial objects on a map and their attribute data values need to be linked uniquely, to avoid spurious results. The reordering between the legacy Columbus data set used the earlier publications and that available for download from the Spacestat website is just one example of a common problem.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

data(oldcol)
rownames(COL.OLD)
data(columbus, package="spData")
rownames(columbus)
get.spChkOption()
oldChk <- set.spChkOption(TRUE)
get.spChkOption()
chkIDs(COL.OLD, nb2listw(COL.nb))
chkIDs(columbus, nb2listw(col.gal.nb))
chkIDs(columbus, nb2listw(COL.nb))
tmp <- try(moran.test(spNamedVec("CRIME", COL.OLD), nb2listw(COL.nb)))
print(tmp)
tmp <- try(moran.test(spNamedVec("CRIME", columbus), nb2listw(col.gal.nb)))
print(tmp)
tmp <- try(moran.test(spNamedVec("CRIME", columbus), nb2listw(COL.nb)))
print(tmp)
set.spChkOption(FALSE)
get.spChkOption()
moran.test(spNamedVec("CRIME", columbus), nb2listw(COL.nb))
tmp <- try(moran.test(spNamedVec("CRIME", columbus), nb2listw(COL.nb),
  spChk=TRUE), silent=TRUE)
print(tmp)
set.spChkOption(oldChk)
get.spChkOption()
Usage

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

Arguments

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

Value

A object of `skater` class with:

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

Author(s)

Renato M. Assuncao and Elias T. Krainski
References


See Also

See Also as mstree

Examples

```r
### loading data
bh <- st_read(system.file("etc/shapes/bhicv.shp", package="spdep")[[1]], quiet=TRUE)
st_crs(bh) <- "+proj=longlat +ellps=WGS84"
### data standardized
dpad <- data.frame(scale(as.data.frame(bh)[,5:8]))

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

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

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

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

### the mstree plot
par(mar=c(0,0,0,0))
plot(st_geometry(bh), border=gray(.5))
plot(mst.bh, coordinates(as(bh, "Spatial")), col=2,
cex.lab=.6, cex.circles=0.035, fg="blue", add=TRUE)

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

### groups size
table(res1$groups)

### the skater plot
opar <- par(mar=c(0,0,0,0))
plot(res1, coordinates(as(bh, "Spatial")), cex.circles=0.035, cex.lab=.7)

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

```r
plot(st_geometry(bh), col=heat.colors(length(res1$edg))[res1$groups])
```

```r
par(opar)
```

### EXPERT OPTIONS

### more one partition

```r
res1b <- skater(res1, dpad, 1)
```

### length groups frequency

```r
table(res1$groups)
table(res1b$groups)
```

### thee groups with minimum population

```r
res2 <- skater(mst.bh[,1:2], dpad, 2, 200000, bh$Pop)
table(res2$groups)
```

### thee groups with minimum number of areas

```r
res3 <- skater(mst.bh[,1:2], dpad, 2, 3, rep(1,nrow(bh)))
table(res3$groups)
```

### thee groups with minimum and maximum number of areas

```r
res4 <- skater(mst.bh[,1:2], dpad, 2, c(20,50), rep(1,nrow(bh)))
table(res4$groups)
```

### if I want to get groups with 20 to 40 elements

```r
res5 <- skater(mst.bh[,1:2], dpad, 2,
c(20,40), rep(1,nrow(bh))) # DON'T MAKE DIVISIONS
table(res5$groups)
```

### In this MST don't have groups with this restrictions

### In this case, first I do one division

### with the minimum criteria

```r
res5a <- skater(mst.bh[,1:2], dpad, 1, 20, rep(1,nrow(bh)))
table(res5a$groups)
```

### and do more one division with the full criteria

```r
res5b <- skater(res5a, dpad, 1, c(20, 40), rep(1,nrow(bh)))
table(res5b$groups)
```

### and do more one division with the full criteria

```r
res5c <- skater(res5b, dpad, 1, c(20, 40), rep(1,nrow(bh)))
table(res5c$groups)
```

### It don't have another division with this criteria

```r
res5d <- skater(res5c, dpad, 1, c(20, 40), rep(1,nrow(bh)))
table(res5d$groups)
```

### Not run:

```r
data(boston, package="spData")
bh.nb <- boston.soi
dpad <- data.frame(scale(boston.c[,]c(7:10))
```
### calculating costs
system.time(lcosts <- nbcosts(bh.nb, dpad))
### making listw
nb.w <- nb2listw(bh.nb, lcosts, style="B")
### find a minimum spanning tree
mst.bh <- mstree(nb.w, 5)
### three groups with no restriction
system.time(res1 <- skater(mst.bh[,1:2], dpad, 2))
library(parallel)
nc <- detectCores(logical=FALSE)
# set nc to 1L here
if (nc > 1L) nc <- 1L
coresOpt <- get.coresOption()
invisible(set.coresOption(nc))
if(!get.mcOption()) {
    # no-op, "snow" parallel calculation not available
    cl <- makeCluster(get.coresOption())
    set.ClusterOption(cl)
}
### calculating costs
system.time(plcosts <- nbcosts(bh.nb, dpad))
all.equal(lcosts, plcosts, check.attributes=FALSE)
### making listw
pnb.w <- nb2listw(bh.nb, plcosts, style="B")
### find a minimum spanning tree
pmst.bh <- mstree(pnb.w, 5)
### three groups with no restriction
system.time(pres1 <- skater(pmst.bh[,1:2], dpad, 2))
if(!get.mcOption()) {
    set.ClusterOption(NULL)
    stopCluster(cl)
}
all.equal(res1, pres1, check.attributes=FALSE)
invisible(set.coresOption(coresOpt))
## End(Not run)
plot(x, main, ylab, ylim, ...)  
## S3 method for class 'spcor'
plot(x, main, ylab, ylim, ...)  
print(x, p.adj.method="none", ...)

Arguments

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

Details

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

Value

returns a list of class spcor:

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

Mantel-Hubert spatial general cross product statistic

---

**Description**

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

sp.mantel.mc(var, listw, nsim, type = "moran", zero.policy = NULL,
alternative = "greater", spChk=NULL, return_boot=FALSE)
## S3 method for class 'mc.sim'
plot(x, xlim, xlab, main, sub, ..., ptype="density")

Arguments

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.
The function retrieves package version and build information.

Usage

```r
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>
Description

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

Usage

```r
lextrB(lw, zero.policy = TRUE, control = list())
lextrW(lw, zero.policy=TRUE, control=list())
lextrS(lw, zero.policy=TRUE, control=list())
#l_max(lw, zero.policy=TRUE, control=list())
griffith_sone(P, Q, type="rook")
subgraph_eigenw(nb, glist=NULL, style="W", zero.policy=NULL, quiet=NULL)
mom_calc(lw, m)
mom_calc_int2(is, m, nb, weights, Card)
stsls(formula, data = list(), listw, zero.policy = NULL,
   na.action = na.fail, robust = FALSE, HC=NULL, legacy=FALSE, W2X = TRUE)
## S3 method for class 'stsls'
impacts(obj, ..., tr, R = NULL, listw = NULL, evaluations=NULL,
   tol = 1e-06, empirical = FALSE, Q=0)
GMerrorsar(formula, data = list(), listw, na.action = na.fail,
   zero.policy = NULL, method="nlminb", arnoldWied=FALSE,
   control = list(), pars=NULL, scaleU=FALSE, verbose=NULL, legacy=FALSE,
   se.lambda=TRUE, returnHcov=FALSE, pWOrder=250, tol.Hcov=1.0e-10)
## S3 method for class 'gmsar'
summary(object, correlation = FALSE, Hausman=FALSE, ...)
GMarginImage(obj, lambda, seq, s2seq)
gstsls(formula, data = list(), listw, listw2 = NULL, na.action = na.fail,
   zero.policy = NULL, pars=NULL, scaleU=FALSE, control = list(),
   verbose=NULL, method="nlminb", robust=FALSE, legacy=FALSE, W2X=TRUE)
## S3 method for class 'gmsar'
impacts(obj, ..., n = NULL, tr = NULL, R = NULL,
   listw = NULL, evaluations=NULL, tol = 1e-06, empirical = FALSE, Q=0)
## S3 method for class 'gmsar'
Hausman.test(object, ..., tol=NULL)
lagmess(formula, data = list(), listw, zero.policy = NULL, na.action = na.fail,
   q = 10, start = -2.5, control=list(), method="BFGS", verbose=NULL,
   use.expm=FALSE)
ME(formula, data=list(), family = gaussian, weights, offset,
   na.action=na.fail, listw=NULL, alpha=0.05, nsim=99, verbose=NULL,
   stdev=FALSE, zero.policy = NULL)
SpatialFiltering(formula, lagformula=NULL, data=list(), na.action=na.fail,
   nb=NULL, glist = NULL, style = "C", zero.policy = NULL, tol = 0.1,
   zerovalue = 1e-04, ExactEV = FALSE, symmetric = TRUE, alpha=NULL,
   ...)
```
alternative="two.sided", verbose=NULL)
LR.sarlm(x, y)
## S3 method for class 'sarlm'
logLik(object, ...)
LR1.sarlm(object)
Wald1.sarlm(object)
## S3 method for class 'sarlm'
Hausman.test(object, ..., tol=NULL)
as.spam.listw(listw)
as_dgRMatrix_listw(listw)
as_dsTMatrix_listw(listw)
as_dsCMatrix_I(n)
as_dsCMatrix_IrW(W, rho)
Jacobian_W(W, rho)
powerWeights(W, rho, order=250, X, tol=.Machine$double.eps^(3/5))
## S3 method for class 'lagImpact'
plot(x, ..., choice="direct", trace=FALSE, density=TRUE)
## S3 method for class 'lagImpact'
print(x, ..., reportQ=NULL)
## S3 method for class 'lagImpact'
summary(object, ..., zstats=FALSE, short=FALSE, reportQ=NULL)
## S3 method for class 'lagImpact'
HPDinterval(obj, prob = 0.95, ..., choice="direct")
intImpacts(rho, beta, P, n, mu, Sigma, irho, drop2beta, bnames, interval,
type, tr, R, listw, evalues, tol, empirical, Q, icerpt, icerpt, p, mess=FALSE,
samples=NULL, zero_fill = NULL, dvars = NULL)
can.be.simmed(listw)
eigenw(listw, quiet=NULL)
similar.listw(listw)
do_ldet(coef, env, which=1)
jacobianSetup(method, env, con, pre_eig=NULL, trs=NULL, interval=NULL, which=1)
cheb_setup(env, q=5, which=1)
mcdet_setup(env, p=16, m=30, which=1)
eigen_setup(env, which=1)
eigen_pre_setup(env, pre_eig, which=1)
spam_setup(env, pivot="MMD", which=1)
spam_update_setup(env, in_coef=0.1, pivot="MMD", which=1)
Matrix_setup(env, Imult, super=as.logical(NA), which=1)
Matrix_J_setup(env, super=FALSE, which=1)
LU_setup(env, which=1)
LU_prepermute_setup(env, coef=0.1, order=FALSE, which=1)
moments_setup(env, trs=NULL, m, p, type="MC", correct=TRUE, trunc=TRUE, eq7=TRUE, which=1)
SE_classic_setup(env, SE_method="LU", p=16, m=30, nrho=200, interp=2000,
interval=c(-1,0.999), SELndet=NULL, which=1)
SE_whichMin_setup(env, SE_method="LU", p=16, m=30, nrho=200, interp=2000,
interval=c(-1,0.999), SELndet=NULL, which=1)
SE_interp_setup(env, SE_method="LU", p=16, m=30, nrho=200,
interval=c(-1,0.999), which=1)
spdep-deprecated

MCMCsamp(object, mcmc = 1L, verbose = NULL, ...)
## S3 method for class 'spautolm'
MCMCsamp(object, mcmc = 1L, verbose = NULL, ...,
burnin = 0L, scale=1, listw, control = list())
## S3 method for class 'sarlm'
MCMCsamp(object, mcmc = 1L, verbose = NULL, ...,
burnin=0L, scale=1, listw, listw2=NULL, control=list())
spautolm(formula, data = list(), listw, weights,
na.action, family = "SAR", method="eigen", verbose = NULL, trs=NULL,
interval=NULL, zero.policy = NULL, tol.solve=.Machine$double.eps,
llprof=NULL, control=list())
## S3 method for class 'spautolm'
sarlm(formula, data = list(), listw, weights,
na.action, family = "SAR", method="eigen", verbose = NULL, trs=NULL,
interval=NULL, zero.policy = NULL, tol.solve=.Machine$double.eps,
llprof=NULL, control=list())
## S3 method for class 'sarlm'
summary(object, correlation = FALSE, adj.se=FALSE,
Nagelkerke=FALSE, ...)
spBreg_sac(formula, data = list(), listw, listw2=NULL, na.action,
Durbin, type, zero.policy=NULL, control=list())
## S3 method for class 'MCMC_sar_g'
impacts(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
## S3 method for class 'MCMC_sem_g'
impacts(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
## S3 method for class 'MCMC_sac_g'
impacts(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
spBreg_err(formula, data = list(), listw, na.action, Durbin, etype,
zero.policy=NULL, control=list())
sarlm(formula, data = list(), listw, weights,
na.action, Durbin, type, method="eigen", quiet=FALSE, 
zero.policy=NULL, interval=NULL, tol.solve=1.0e-10, trs=NULL, control=list())
## S3 method for class 'sarlm'
predict(object, newdata, listw, zero.policy=NULL, ...)
lmSLX(formula, data = list(), listw, na.action, weights=NULL,
Durbin=TRUE, zero.policy=NULL)
## S3 method for class 'SLX'
impacts(obj, ...)
create WX(x, listw, zero.policy=NULL, prefix="")
## S3 method for class 'sarlm'
anova(object, ...)
bptest.sarlm(object, varformula=NULL, studentize = TRUE, data=list())
errorsarlm(formula, data=list(), listw, na.action, weights=NULL,
Durbin, etype, method="eigen", quiet=FALSE, zero.policy=NULL, 
interval = NULL, tol.solve=1.0e-10, trs=NULL, control=list())
## S3 method for class 'sarlm'
impacts(obj, ..., tr = NULL, listw = NULL, evalues=NULL,
useHESS = NULL, tol = 1e-06, empirical = FALSE, Q=NULL)
lagsarlm(formula, data = list(), listw,
na.action, Durbin, type, method="eigen", quiet=NULL,
zero.policy=NULL, interval=NULL, tol.solve=1.0e-10, trs=NULL,
control=list())
## S3 method for class 'sarlm'
predict(object, newdata = NULL, listw = NULL, pred.type = "TS", all.data = FALSE,
spdep-deprecated

zero.policy = NULL, legacy = TRUE, legacy.mixed = FALSE, power = NULL, order = 250,
tol = .Machine$double.eps^(3/5), spChk = NULL, ...
## S3 method for class 'sarlm.pred'
print(x, ...)
## S3 method for class 'sarlm.pred'
print(x, ...)
## S3 method for class 'sarlm'
residuals(object, ...)
## S3 method for class 'sarlm'
deviance(object, ...)
## S3 method for class 'sarlm'
coef(object, ...)
## S3 method for class 'sarlm'
vcov(object, ...)
## S3 method for class 'sarlm'
fitted(object, ...)
sacsarlm(formula, data = list(), listw, listw2 = NULL, ha.action, Durbin, type,
method = "eigen", quiet = NULL, zero.policy = NULL, tol.solve = 1e-10,
llprof=NULL, interval1=NULL, interval2=NULL, trs1=NULL, trs2=NULL,
control = list())
## S3 method for class 'sarlm'
summary(object, correlation = FALSE, Nagelkerke = FALSE, Hausman=FALSE, adj.se=FALSE, ...)
## S3 method for class 'summary.sarlm'
print(x, digits = max(5, .Options$digits - 3),
signif.stars = FALSE, ...)
trW(W=NULL, m = 30, p = 16, type = "mult", listw=NULL, momentsSymmetry=TRUE)

Arguments

lw a binary symmetric listw object from, for example, nb2listw with style “B” for lextrB, style “W” for lextrW and style “S” for lextrS; for l_max, the object may be asymmetric and does not have to be binary

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

control a list of control arguments

quiet default NULL, use global !verbose option value; set to FALSE for short summary

P number of columns in the grid (number of units in a horizontal axis direction)

Q number of rows in the grid (number of units in a vertical axis direction.)

type “rook” or “queen”

nb an object of class nb

glist list of general weights corresponding to neighbours

style style can take values “W”, “B”, “C”, “U”, “minmax” and “S”
m The number of powers; must be an even number for 'type'="moments" (default changed from 100 to 30 (2010-11-17))
is (used internally only in `mom_calc_int2` for 'type'="moments" on a cluster)
weights (used internally only in `mom_calc_int2` for 'type'="moments" on a cluster)
Card (used internally only in `mom_calc_int2` for 'type'="moments" on a cluster)
formula a symbolic description of the model to be fit. The details of model specification are given for `lm()`
data an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
listw a `listw` object created for example by `nb2listw`
na.action a function (default `na.fail`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
robust default FALSE, if TRUE, apply a heteroskedasticity correction to the coefficients covariances
HC default NULL, if robust is TRUE, assigned "HC0", may take values "HC0" or "HC1" for White estimates or MacKinnon-White estimates respectively
legacy the argument chooses between two implementations of the robustness correction: default FALSE - use the estimate of Omega only in the White consistent estimator of the variance-covariance matrix, if TRUE, use the original implementation which runs a GLS using the estimate of Omega, and yields different coefficient estimates as well - see example below
W2X default TRUE, if FALSE only WX are used as instruments in the spatial two stage least squares; until release 0.4-60, only WX were used - see example below
obj A spatial regression object created by `lagsarlm`, `lagmess` or by `lmSLX`; 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, and must be row-standardised
evalues vector of eigenvalues of spatial weights matrix for impacts calculations
n defaults to `length(obj$Residuals)` in `gmsar` objects it may be used in panel settings to compute the impacts for cross-sectional weights only, suggested by Angela Parenti
R If given, simulations are used to compute distributions for the impact measures, returned as `mcmc` objects; the objects are used for convenience but are not output by an MCMC process
tol Argument passed to `mvnorm`: tolerance (relative to largest variance) for numerical lack of positive-definiteness in the coefficient covariance matrix
Argument passed to `mvrnorm` (default FALSE): if true, the coefficients and their covariance matrix specify the empirical not population mean and covariance matrix.

A `listw` object created for example by `nb2listw`, if not given, set to the same spatial weights as the `listw` argument.

Starting values for $\lambda$ and $\sigma^2$ for GMM optimisation, if missing (default), approximated from initial 2sls model as the autocorrelation coefficient corrected for weights style and model sigma squared.

Default FALSE: scale the OLS residuals before computing the moment matrices; only used if the `pars` argument is missing.

Default “nlminb”, or optionally a method passed to `optim` to use an alternative optimiser.

Default FALSE, return the Vo matrix for a spatial Hausman test.

The tolerance for computing the Vo matrix (default=1.0e-10).

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

If given, an increasing sequence of lambda values for gridding.

If given, an increasing sequence of sigma squared values for gridding.

Gmm object from `GMerrorsar`.

Logical; (default=FALSE), TRUE not available.

If TRUE, the results of the Hausman test for error models are reported.


Default NULL, use global option value; if TRUE, reports function values during optimization.

Default 10; number of powers of the spatial weights to use.

Starting value for numerical optimization, should be a small negative number.

Default FALSE; if TRUE use `expm::expAtv` instead of a truncated power series of W.

A description of the error distribution and link function to be used in the model.

This can be used to specify an a priori known component to be included in the linear predictor during fitting.

Used as a stopping rule to choose all eigenvectors up to and including the one with a p-value exceeding alpha.

Number of permutations for permutation bootstrap for finding p-values.

If TRUE, p-value calculated from bootstrap permutation standard deviate using `pnorm` with alternative=”greater”, if FALSE the Hope-type p-value.

An extra one-sided formula to be used when a spatial lag representation is desired; the intercept is excluded within the function if present because it is part of the formula argument, but excluding it explicitly in the `lagformula` argument in the presence of factors generates a collinear model matrix.
zerovalue eigenvectors with eigenvalues of an absolute value smaller than zerovalue will be excluded in eigenvector search

ExactEV Set ExactEV=TRUE to use exact expectations and variances rather than the expectation and variance of Moran’s I from the previous iteration, default FALSE

symmetric Should the spatial weights matrix be forced to symmetry, default TRUE

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

x a logLik object or an object for which a logLik() function exists

y a logLik object or an object for which a logLik() function exists

W a dsTMatrix object created using as_dsTMatrix_listw from a symmetric listw object

rho spatial regression coefficient

order Power series maximum limit

X A numerical matrix

choice One of three impacts: direct, indirect, or total

trace Argument passed to plot.mcmc: plot trace plots

density Argument passed to plot.mcmc: plot density plots

prob Argument passed to HPDinterval.mcmc: a numeric scalar in the interval (0,1) giving the target probability content of the intervals

beta, mu, Sigma, irho, drop2beta, bnames, interval, icept, iicept, p, mess, samples, zero_fill, dvars internal arguments shared inside impacts methods

reportQ default NULL; if TRUE and Q given as an argument to impacts, report impact components

zstats default FALSE, if TRUE, also return z-values and p-values for the impacts based on the simulations

short default FALSE, if TRUE passed to the print summary method to omit printing of the mcmc summaries

coef spatial coefficient value

eqv environment containing pre-computed objects, fixed after assignment in setup functions

which default 1; if 2, use second listw object

con control list passed from model fitting function and parsed in jacobianSetup to set environment variables for method-specific setup

pre_eig pre-computed eigenvalues of length n

pivot default “MMD”, may also be “RCM” for Cholesky decomposition using 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 maximum 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, trs1, trs2

A numeric vector of m traces, as from trW.

correct

default TRUE: use Smirnov correction term, see trW.

trunc

default TRUE: truncate Smirnov correction term, see trW.

eq7

default TRUEuse equation 7 in Smirnov and Anselin (2009), if FALSE no unit root correction.

SE_method

default “LU”, alternatively “MC”; underlying lndet method to use for generating SE toolbox emulation grid.

nrho

default 200, number of lndet values in first stage SE toolbox emulation grid.

interval1, interval2

default c(-1,0.999) if interval argument NULL, bounds for SE toolbox emulation grid.

interp

default 2000, number of lndet values to interpolate in second stage SE toolbox emulation grid.

SElndet

default NULL, used to pass a pre-computed two-column matrix of coefficient values and corresponding interpolated lndet values.

mcmc

The number of MCMC iterations after burnin.

burnin

The number of burn-in iterations for the sampler.

scale

da positive scale parameter.

tol.solve

the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to solve() (default=double precision machine tolerance). Errors in solve() may constitute indications of poorly scaled variables: if the variables have scales differing much from the autoregressive coefficient, the values in this matrix may be very different in scale, and inverting such a matrix is analytically possible by definition, but numerically unstable; rescaling the RHS variables alleviates this better than setting tol.solve to a very small value.

llprof

default NULL, can either be an integer, to divide the feasible range into llprof points, or a sequence of spatial coefficient values, at which to evaluate the likelihood function.

adj.se

if TRUE, adjust the coefficient standard errors for the number of fitted coefficients.

Nagelkerke

if TRUE, the Nagelkerke pseudo R-squared is reported.

Durbin

default FALSE (spatial lag model); if TRUE, full spatial Durbin model; if a formula object, the subset of explanatory variables to lag.

etype

(use the ‘Durbin=’ argument - retained for backwards compatibility only) default "error", may be set to "emixed" to include the spatially lagged independent variables added to X; when "emixed", the lagged intercept is dropped for spatial weights style "W", that is row-standardised weights, but otherwise included.
newdata data frame in which to predict — if NULL, predictions are for the data on which the model was fitted. Should have row names corresponding to region.id. If row names are exactly the same than the ones used for training, it uses in-sample predictors for forecast. See ‘Details’

prefix default empty string, may be “lag” in some cases

varformula a formula describing only the potential explanatory variables for the variance (no dependent variable needed). By default the same explanatory variables are taken as in the main regression model

studentize logical. If set to TRUE Koenker’s studentized version of the test statistic will be used.

useHESS, pred.type, all.data, legacy.mixed, power, spChk, digits, signif.stars other arguments in deprecated functions

momentsSymmetry default TRUE; assert Smirnov/Anselin symmetry assumption

Details

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

See Also

  Deprecated

Examples

  ## Not run:
  data(boston, package="spData")
  ab.listb <- nb2listw(boston.soi, style="B")
  er <- range(eigenw(ab.listb))
  er
  res_1 <- lextkB(ab.listb)
  c(res_1)
  run <- FALSE
  if (require("RSpectra", quietly=TRUE)) run <- TRUE
  if (run) {
    B <- as(ab.listb, "CsparseMatrix")
    eigs(B, k=1, which="SR")$values
  }
  if (run) {
    eigs(B, k=1, which="LR")$values
  }
  k5 <- knn2nb(knearneigh(boston.utm, k=5))
  c(l_max(nb2listw(k5, style="B")))
  max(Re(eigenw(nb2listw(k5, style="B"))))
  c(l_max(nb2listw(k5, style="C")))
  max(Re(eigenw(nb2listw(k5, style="C"))))
  ab.listw <- nb2listw(boston.soi, style="W")
  er <- range(eigenw(similar.listw(ab.listw)))
  er
res_1 <- lextrW(ab.listw)
c(res_1)
if (run) {
  B <- as(similar.listw(ab.listw), "CsparseMatrix")
  eigs(B, k=1, which="SR")$values
}
if (run) {
  eigs(B, k=1, which="LR")$values
}
ab.listw <- nb2listw(boston.soi, style="S")
er <- range(eigenw(similar.listw(ab.listw)))
er
res_1 <- lextrS(ab.listw)
c(res_1)
if (run) {
  B <- as(similar.listw(ab.listw), "CsparseMatrix")
  eigs(B, k=1, which="SR")$values
}
if (run) {
  eigs(B, k=1, which="LR")$values
}
rg <- cell2nb(ncol=7, nrow=7, type="rook")
rg_eig <- eigenw(nb2listw(rg, style="B"))
rg_GS <- griffith_sone(P=7, Q=7, type="rook")
all.equal(rg_eig, rg_GS)
# subgraphs
data(oldcol)
crds <- cbind(COL.OLD$X, COL.OLD$Y)
k3 <- knn2nb(knearneigh(crds, k=3))
nc <- n.comp.nb(k3)
nc$nc
table(nc$comp.id)
k3eig <- eigenw(nb2listw(k3, style="W"))
k3eigSG <- subgraph_eigenw(k3, style="W")
all.equal(sort(k3eig), k3eigSG)
data(oldcol)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
summary(COL.lag.eig, correlation=TRUE)
COL.lag.stsls <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb))
summary(COL.lag.stsls, correlation=TRUE)
COL.lag.stslsW <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb), W2X=FALSE)
summary(COL.lag.stslsW, correlation=TRUE)
COL.lag.stslsR <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb), robust=TRUE, W2X=FALSE)
summary(COL.lag.stslsR, correlation=TRUE)
COL.lag.stslsRl <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, nb2listw(COL.nb), robust=TRUE, legacy=TRUE, W2X=FALSE)
summary(COL.lag.stslsRl, correlation=TRUE)
data(boston, package="spData")
gp2a <- stsls(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi))
```r
spdep-deprecated

summary(gp2a)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
listw <- nb2listw(col.gal.nb)
ev <- eigenw(listw)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
lobj1 <- stsls(CRIME ~ INC + HOVAL, columbus, listw)
loobj1 <- impacts(lobj1, R=200, tr=trMatc)
summary(loobj1, zstats=TRUE, short=TRUE)
loobj2 <- impacts(lobj1, R=200, evals=ev)
summary(loobj2, zstats=TRUE, short=TRUE)
library(coda)
HPDinterval(loobj1)
lobj1r <- stsls(CRIME ~ INC + HOVAL, columbus, listw, robust=TRUE)
loobj1r <- impacts(lobj1r, tr=trMatc, R=200)
summary(loobj1r, zstats=TRUE, short=TRUE)
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="W"), method="eigen")
summary(COL.errW.eig, Hausman=TRUE)
COL.errW.GM <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="W"), returnHcov=TRUE)
summary(COL.errW.GM, Hausman=TRUE)
aa <- GMargminImage(COL.errW.GM)
levs <- quantile(aa$z, seq(0, 1, 1/12))
image(aa, breaks=levs, xlab="lambda", ylab="s2")
points(COL.errW.GM$lambda, COL.errW.GM$s2, pch=3, lwd=2)
contour(aa, levels=signif(levs, 4), add=TRUE)
COL.errW.GM1 <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="W"))
summary(COL.errW.GM1)
nadata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[[1]], quiet=TRUE)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]]))
suppressMessages(ID <- as.character(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]]))[-1]))
identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEY), 2, 10))
yadjlw <- mat2listw(nyadjmat, as.character(nydata$AREAKEY))
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
listw=listw_NY, family="SAR", method="eigen")
summary(esar1f)
esar1gm <- GMerrorsar(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
data=nydata, listw=listw_NY)
summary(esar1gm)
esar1gm1 <- GMerrorsar(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
data=nydata, listw=listw_NY, method="Nelder-Mead")
summary(esar1gm1)
data(baltimore, package="spData")
baltimore$AGE <- ifelse(baltimore$AGE < 1, 1, baltimore$AGE)
lw <- nb2listw(knn2nb(kearneigh(cbind(baltimore$X, baltimore$Y), k=7)))
obj1 <- lm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT),
```

data=baltimore)
lm.moran.test(obj1, lw)
lm.LMtests(obj1, lw, test="all")
summary(obj2)
summary(obj2a)
obj3 <- lagsarlm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw)
summary(obj3)
data(boston, package="spData")
lw <- nb2listw(boston.soi)
gp2 <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) +
          AGE + log(DIS) + log(RAD) + TAX + PTRAIO + 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 + PTRAIO + B + log(LSTAT),
data=boston.c, lw)
summary(gp2a)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)
lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL, data=columbus,
                          nb=col.gal.nb, style="W", alpha=0.1, verbose=TRUE)
lagcol
lmag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
anova(lmbase, lmag)
set.seed(123)
lagcol1 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
             listw=nb2listw(col.gal.nb), alpha=0.1, verbose=TRUE)
lagcol1
lmag1 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol1), data=columbus)
anova(lmbase, lmag1)
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
lmag2 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol2), data=columbus)
anova(lmbase, lmag2)
NA.columbus <- columbus
COL.ME.NA <- ME(CRIME ~ INC + HOVAL, data=NA.columbus, family="gaussian",
                listw=nb2listw(col.gal.nb), alpha=0.1, stdev=TRUE, verbose=TRUE,
                na.action=na.exclude)
COL.ME.NA$na.action
summary(lm(CRIME ~ INC + HOVAL + fitted(COL.ME.NA), data=NA.columbus,
            na.action=na.exclude))
#nc.sids <- st_read(system.file("shapes/sids.shp", package="spData")[[1]], quiet=TRUE)
#rn <- as.character(nc.sids$FIPS)
#ncCC89_nb <- read.gal(system.file("weights/ncCC89.gal", package="spData")[[1]],
#                # region.id=rn)
#ncCR85_nb <- read.gal(system.file("weights/ncCR85.gal", package="spData")[[1]],


# region.id=rn
# glmbase <- glm(SID74 ~ 1, data=nc.sids, offset=log(BIR74),
# family="poisson")
# set.seed(123)
# MEpois1 <- ME(SID74 ~ 1, data=nc.sids, offset=log(BIR74),
# family="poisson", listw=nb2listw(ncCR85_nb, style="B"), alpha=0.2, verbose=TRUE)
# MEpois1
# glmME <- glm(SID74 ~ 1 + fitted(MEpois1), data=nc.sids, offset=log(BIR74),
# family="poisson")
# anova(glmME, test="Chisq")
# anova(glmbase, glmME, test="Chisq")
data(hopkins, package="spData")
hopkins_part <- hopkins[21:36,36:21]
hopkins_part[which(hopkins_part > 0, arr.ind=TRUE)] <- 1
hopkins.rook.nb <- cell2nb(16, 16, type="rook")
glmbase <- glm(c(hopkins_part) ~ 1, family="binomial")
set.seed(123)
MEbinom1 <- ME(c(hopkins_part) ~ 1, family="binomial",
listw=nb2listw(hopkins.rook.nb, style="B"), alpha=0.2, verbose=TRUE)
glmME <- glm(c(hopkins_part) ~ 1 + fitted(MEbinom1), family="binomial")
anova(glmME, test="Chisq")
anova(glmbase, glmME, test="Chisq")
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)
sarcol <- SpatialFiltering(CRIME ~ INC + HOVAL, data=columbus, nb=col.gal.nb, style="W", ExactEV=TRUE)
sarcol
lmsar <- lm(CRIME ~ INC + HOVAL + fitted(sarcol), data=columbus)
lmsar
anova(lmbase, lmsar)
lm.morantest(lmsar, nb2listw(col.gal.nb))
lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL - 1, data=columbus, nb=col.gal.nb, style="W")
lagcol
lmlag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
lmlag
anova(lmbase, lmlag)
lm.morantest(lmlag, nb2listw(col.gal.nb))
NA.columbus <- columbus
COL.SF.NA <- SpatialFiltering(CRIME ~ INC + HOVAL, data=NA.columbus, nb=col.gal.nb, style="W", na.action=na.exclude)
COL.SF.NA$na.action
summary(lm(CRIME ~ INC + HOVAL + fitted(COL.SF.NA), data=NA.columbus, na.action=na.exclude))
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[1], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
mixed <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb), type="mixed")
error <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
LR.sarlm(mixed, error)
Hausman.test(error)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
col.listw <- nb2listw(col.gal.nb)
if (require("spam", quietly=TRUE)) {
col.sp <- as.spam.listw(col.listw)
str(col.sp)
}
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]])[-1]))
nyadjlw <- mat2listw(nyadjmat)
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
library(Matrix)
W_C <- as(listw_NY, "CsparseMatrix")
W_R <- as(listw_NY, "RsparseMatrix")
W_S <- as(listw_NY, "symmetricMatrix")
n <- nrow(W_S)
I <- Diagonal(n)
rho <- 0.1
c(determinant(I - rho * W_S, logarithm=TRUE)$modulus)
sum(log(1 - rho * eigenw(listw_NY)))
W <- as(W_S, "CsparseMatrix")
set.seed(1)
x <- matrix(rnorm(dim(W)[1]), ncol=1)
system.time(ee <- powerWeights(W, rho=0.3, X=x))
#cases for intrinsically asymmetric weights
crs <- cbind(COL.OLD$X, COL.OLD$Y)
k3 <- knn2nb(knearneigh(crs, k=3))
is.symmetric.nb(k3)
k3eig <- eigenw(nb2listw(k3, style="W"))
is.complex(k3eig)
rho <- 0.5
Jc <- sum(log(1 - rho * k3eig))
# complex eigenvalue Jacobian
Jc
W <- as(nb2listw(k3, style="W"), "CsparseMatrix")
I <- diag(length(k3))
Jl <- sum(log(abs(diag(slot(lu(I - rho * W), "U")))))
# LU Jacobian equals complex eigenvalue Jacobian
Jl
all.equal(Re(Jc), Jl)
# wrong value if only real part used
Jr <- sum(log(1 - rho * Re(k3eig)))
Jr
all.equal(Jr, Jl)
# construction of Jacobian from complex conjugate pairs (Jan Hauke)
Rev <- Re(k3eig)[which(Im(k3eig) == 0)]
# real eigenvalues
Cev <- k3eig[which(Im(k3eig) != 0)]
pCev <- Cev[Im(pCev) > 0]
# separate complex conjugate pairs
RpCev <- Re(pCev)
IpCev <- Im(pCev)
# reassemble Jacobian
Jc1 <- sum(log(1 - rho*Rev)) + sum(log((1 - rho * RpCev)^2 + (rho^2)*(IpCev^2)))
all.equal(Re(Jc), Jc1)
# impact of omitted complex part term in real part only Jacobian
Jc2 <- sum(log(1 - rho*Rev)) + sum(log((1 - rho * RpCev)^2))
all.equal(Jr, Jc2)
# trace of asymmetric (WW) and crossprod of complex eigenvalues for APLE
sum(diag(W %*% W))
crossprod(k3eig)
# analytical regular grid eigenvalues
run <- FALSE
if (require("RSpectra", quietly=TRUE)) run <- TRUE
if (run) {
  rg <- cell2nb(ncol=7, nrow=7, type="rook")
  B <- as(nb2listw(rg, style="B"), "CsparseMatrix")
  res1 <- eigs(B, k=1, which="LR")$values
  resn <- eigs(B, k=1, which="SR")$values
  print(Re(c(resn, res1)))
}
if (run) {
  rg_eig <- eigenw(nb2listw(rg, style="B"))
  print(all.equal(range(Re(rg_eig)), c(resn, res1)))
}
if (run) {
  lw <- nb2listw(rg, style="W")
  rg_eig <- eigenw(similar.listw(lw))
  print(range(Re(rg_eig)))
}
if (run) {
\[ W \leftarrow \text{as}(lw, \text{"CsparseMatrix"}) \]
\[
\text{print}(\text{Re}(\text{c}(\text{eigs}(W, k=1, \text{which}=\text{"SR"})$values, \text{eigs}(W, k=1, \text{which}=\text{"LR"})$values)))}
\]
data(oldcol)
COL.W \leftarrow \text{nb2listw(COL.nb, style="W")}
COL.S \leftarrow \text{nb2listw(COL.nb, style="S")}
\[
\text{sum}(\log(1 - 0.5 \times \text{eigenw}(\text{COL.W})))
\]
\[
\text{sum}(\log(1 - 0.5 \times \text{eigenw}(\text{similar.listw}(\text{COL.W}))))
\]
\[ W.J \leftarrow \text{as}(\text{as_dsTMatrix_listw}(\text{similar.listw}(\text{COL.W}), \text{"CsparseMatrix"})) \]
\[ I \leftarrow \text{as_dsCMatrix_I}(\text{dim}(W.J)[1]) \]
\[
c(\text{determinant}(I - 0.5 \times W.J, \text{logarithm=TRUE})$\text{modulus})
\]
\[
\text{sum}(\log(1 - 0.5 \times \text{eigenw}(\text{COL.S})))
\]
\[
\text{sum}(\log(1 - 0.5 \times \text{eigenw}(\text{similar.listw}(\text{COL.S}))))
\]
\[ W.J \leftarrow \text{as}(\text{as_dsTMatrix_listw}(\text{similar.listw}(\text{COL.S}), \text{"CsparseMatrix"})) \]
\[
c(\text{determinant}(I - 0.5 \times W.J, \text{logarithm=TRUE})$\text{modulus})
\]
data(boston, package=\text{spData})
lw \leftarrow \text{nb2listw(boston.soi)}
can.sim \leftarrow \text{spdep:::can.be.simmed(lw)}
env \leftarrow \text{new.env(parent=globalenv())}
\text{assign}(\text{"listw"}, lw, \text{envir}=env)
\text{assign}(\text{"can.sim"}, can.sim, \text{envir}=env)
\text{assign}(\text{"similar"}, \text{FALSE}, \text{envir}=env)
\text{assign}(\text{"verbose"}, \text{FALSE}, \text{envir}=env)
\text{assign}(\text{"family"}, \text{"SAR"}, \text{envir}=env)
\text{eigen_setup(env)}
\text{get}(\text{"similar"}, \text{envir}=env)
\text{do_ldet}(0.5, env)
\text{rm(env)}

env \leftarrow \text{new.env(parent=globalenv())}
\text{assign}(\text{"listw"}, lw, \text{envir}=env)
\text{assign}(\text{"can.sim"}, can.sim, \text{envir}=env)
\text{assign}(\text{"similar"}, \text{FALSE}, \text{envir}=env)
\text{assign}(\text{"verbose"}, \text{FALSE}, \text{envir}=env)
\text{assign}(\text{"family"}, \text{"SAR"}, \text{envir}=env)
\text{assign}(\text{"n"}, \text{length}(\text{boston.soi}), \text{envir}=env)
\text{eigen_pre_setup(env, pre_eig=\text{eigenw}(\text{similar.listw}(\text{lw})))}
\text{do_ldet}(0.5, env)
\text{rm(env)}

env \leftarrow \text{new.env(parent=globalenv())}
\text{assign}(\text{"listw"}, lw, \text{envir}=env)
\text{assign}(\text{"can.sim"}, can.sim, \text{envir}=env)
\text{assign}(\text{"similar"}, \text{FALSE}, \text{envir}=env)
\text{assign}(\text{"family"}, \text{"SAR"}, \text{envir}=env)
\text{assign}(\text{"n"}, \text{length}(\text{boston.soi}), \text{envir}=env)
\text{Matrix_setup(env, Imult=2, super=FALSE)}
\text{get}(\text{"similar"}, \text{envir}=env)
\text{do_ldet}(0.5, env)
\text{rm(env)}

\text{if (require(\"spam\", quietly=TRUE)) \{ }
\text{env \leftarrow \text{new.env(parent=globalenv())}}
\text{assign}(\text{"listw"}, lw, \text{envir}=env)
\text{assign}(\text{"n"}, \text{length}(\text{boston.soi}), \text{envir}=env)
\text{assign}(\text{"can.sim"}, can.sim, \text{envir}=env)\text{\}}
assign("similar", FALSE, 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)
LU_prepermutate_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("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
set.seed(12345)
mcdet_setup(env, p=16, m=30)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[[1]], quiet=TRUE)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]][-1])))
suppressMessages(ID <- as.character(names(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]][-1])))
identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEY), 2, 10))
nyadjlw <- mat2listw(nyadjmat)
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, listw=listw_NY, family="SAR", method="eigen")
summary(esar1f)
res <- MCMCsamp(esar1f, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
#esar1fw <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, weights=POP8, family="SAR", method="eigen")
#summary(esar1fw)
#res <- MCMCsamp(esar1fw, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
ecar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, family="CAR", method="eigen")
summary(ecar1f)
res <- MCMCsamp(ecar1f, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
#esar1fw <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, weights=POP8, family="SAR", method="eigen")
#summary(esar1fw)
#res <- MCMCsamp(esar1fw, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
#ecar1fw <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, weights=POP8, family="CAR", method="eigen")
#summary(ecar1fw)
#res <- MCMCsamp(ecar1fw, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
esar0 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY)
summary(esar0)
res <- MCMCsamp(esar0, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
#esar0w <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, weights=POP8)
#summary(esar0w)
#res <- MCMCsamp(esar0w, mcmc=5000, burnin=500, listw=listw_NY)
#summary(res)
esar1 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, etype="emixed")
summary(esar1)
res <- MCMCsamp(esar1, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
lsar0 <- lagsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY)
summary(lsar0)
res <- MCMCsamp(lsar0, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
lsar1 <- lagsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY, type="mixed")
summary(lsar1)
res <- MCMCsamp(lsar1, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
ssar0 <- sacsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, # listw=listw_NY)
summary(ssar0)
res <- MCMCsamp(ssar0, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
ssar1 <- sacsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, 
                 listw=listw_NY, type="sacmixed")
summary(ssar1)
res <- MCMCsamp(ssar1, mcmc=5000, burnin=500, listw=listw_NY)
summary(res)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[[1]], quiet=TRUE)
lm0 <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata)
summary(lm0)
lm0w <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, weights=POP8)
summary(lm0w)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]])[-1]))
suppressMessages(ID <- as.character(names(foreign::read.dbf(system.file("misc/nyadjwts.dbf", package="spData")[[1]])[-1])))
identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEY), 2, 10))
nyadjlw <- mat2listw(nyadjmat, as.character(nydata$AREAKEY))
listw_NY <- nb2listw(nyadjlw$neighbours, style="B")
esar0 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, 
                    listw=listw_NY)
summary(esar0)
system.time(esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, 
                              data=nydata, listw=listw_NY, family="SAR", method="eigen")
             res <- summary(esar1f)
print(res)
sqrt(diag(res$res$resvar))
sqrt(diag(esar1f$fit$imat)*esar1f$fit$s2)
sqrt(diag(esar1f$fdHess))
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, 
                              data=nydata, listw=listw_NY, family="SAR", method="Matrix")
             summary(esar1M)
#esar1w <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, 
#                   data=nydata, listw=listw_NY, weights=POP8, family="SAR", method="eigen")
#summary(esar1w)
#system.time(esar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, 
#                              # data=nydata, listw=listw_NY, weights=POP8, family="SAR", method="Matrix")
#summary(esar1wM)
#esar1wu <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, 
#                   # listw=listw_NY, weights=POP8, family="SAR", method="LU")
#summary(esar1wu)
#esar1wch <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, 
#                    # listw=listw_NY, weights=POP8, family="SAR", method="Chebyshev")
#summary(esar1wch)
ecar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, 
                   listw=listw_NY, family="CAR", method="eigen")
summary(ecar1f)
system.time(ecar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, 
                           # data=nydata, listw=listw_NY, family="CAR", method="Matrix")
          summary(ecar1M)
#ecar1w <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
# listw=listw_NY, weights=nydata$POP8, family="CAR", method="eigen"
#summary(ecar1wf)
#system.time(ecar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, # data=nydata, listw=listw_NY, weights=POP8, family="CAR", method="Matrix"))
#summary(ecar1wM)
nc.sids <- st_read(system.file("shapes/sids.shp", package="spData")[1], quiet=TRUE)
ft.SID74 <- sqrt(1000)*(sqrt(nc.sids$SID74/nc.sids$BIR74) +
                    sqrt((nc.sids$SID74+1)/nc.sids$BIR74))
lm_nc <- lm(ft.SID74 ~ 1)
sids.nhbr30 <- dnearneigh(cbind(nc.sids$east, nc.sids$north), 0, 30, row.names=row.names(nc.sids))
sids.nhbr30.dist <- nbdists(sids.nhbr30, cbind(nc.sids$east, nc.sids$north))
sids.nhbbr <- listw2sn(nb2listw(sids.nhbr30, glist=sids.nhbr30.dist, style="B", zero.policy=TRUE))
dij <- sids.nhbbr[,3]
n <- nc.sids$BIR74
e1 <- min(dij)/dij
e2 <- sqrt(n[sids.nhbbr$to]/n[sids.nhbbr$from])
sids.nhbbr$weights <- e1*e2
tsids.nhbbr.listw <- sn2listw(sids.nhbbr)
both <- factor(paste(nc.sids$L_id, nc.sids$M_id, sep=":"))
ft.NWBIR74 <- sqrt(1000)*(sqrt(nc.sids$NWBIR74/nc.sids$BIR74) +
                        sqrt((nc.sids$NWBIR74+1)/nc.sids$BIR74))
mdata <- data.frame(both, ft.NWBIR74, ft.SID74, BIR74=nc.sids$BIR74)
outl <- which.max(rstandard(lm_nc))
as.character(nc.sids$NAME[outl])
mdata.4 <- mdata[-outl,]
W <- listw2mat(sids.nhbbr.listw)
W.4 <- W[-outl, -outl]
sids.nhbbr.listw.4 <- mat2listw(W.4)
esarI <- errorsarlm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbbr.listw,
                    zero.policy=TRUE)
summary(esarI)
esarIa <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbbr.listw,
                   family="SAR")
summary(esarIa)
esarIV <- errorsarlm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbbr.listw,
                     zero.policy=TRUE)
summary(esarIV)
esarIVA <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbbr.listw,
                    family="SAR")
summary(esarIVA)
#esarIaw <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbbr.listw,
#                   weights=BIr74, family="SAR")
#summary(esarIaw)
#esarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata, listw=sids.nhbbr.listw,
#                    weights=BIr74, family="SAR")
#summary(esarIIaw)
#esarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata,
#                     listw=sids.nhbbr.listw, weights=BIr74, family="SAR")
#summary(esarIVaw)
#ecarIaw <- spautolm(ft.SID74 ~ 1, data=mdata.4, listw=sids.nhbbr.listw.4,
#                    weights=BIr74, family="CAR")
#summary(ecarIaw)
#ecarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata.4,
# listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
#summary(ecarIIaw)
#ecarIVaw <- spautolm(Ft.SID74 ~ ft.NWBIR74, data=mdata.4,
# listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
#summary(ecarIVaw)
#nc.sids$fitIV <- append(fitted.values(ecarIVaw), NA, outl-1)
#plot(nc.sids[, "fitIV"], nbreaks=12) # Cressie 1993, p. 565
data(oldcol)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"))
summary(COL.errW.eig)
COL.errW.sar <- spautolm(CRIME ~ INC + HOVAL, data=COL.OLD,
 nb2listw(COL.nb, style="W"))
summary(COL.errW.sar)
data(boston, package="spData")
gp1 <- spautolm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2)
 + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, nb2listw(boston.soi), family="SMA")
summary(gp1)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
ev <- eigenw(listw)
COL.sacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
 control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.sacW.eig)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
set.seed(1)
summary(impacts(COL.sacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
library(coda)
set.seed(1)
COL.sacW.B0 <- spBreg_sac(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
 Durbin=FALSE, control=list(ndraw=1500L, nomit=500L))
print(summary(COL.sacW.B0))
print(summary(impacts(COL.sacW.B0, tr=trMatc), zstats=TRUE, short=TRUE))
set.seed(1)
COL.sacW.B1 <- spBreg_sac(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
 Durbin=TRUE, control=list(ndraw=1500L, nomit=500L))
print(summary(COL.sacW.B1))
print(summary(impacts(COL.sacW.B1, tr=trMatc), zstats=TRUE, short=TRUE))
COL.msacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
 type="sacmixed", control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW.eig)
set.seed(1)
summary(impacts(COL.msacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW1.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
 Durbin=TRUE, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW1.eig)
set.seed(1)
summary(impacts(COL.msacW1.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW2.eig <- sacsarlm(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
 listw, Durbin= ~ INC, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW2.eig)
library(coda)
set.seed(1)
data(oldcol)
lw <- nb2listw(COL.nb, style="W")
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
Durbin=TRUE)
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
Durbin= ~ INC)
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
Durbin=~INC, control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
COL.lag.Bayes <- spBreg_lag(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw)
summary(COL.lag.Bayes)
summary(impacts(COL.lag.Bayes, tr=trMatc), short=TRUE, zstats=TRUE)
summary(impacts(COL.lag.Bayes, evalues=ev), short=TRUE, zstats=TRUE)
set.seed(1)
COL.D0.Bayes <- spBreg_lag(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw, Durbin=TRUE)
summary(COL.D0.Bayes)
summary(impacts(COL.D0.Bayes, tr=trMatc), short=TRUE, zstats=TRUE)
set.seed(1)
COL.D1.Bayes <- spBreg_lag(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD, listw=listw, Durbin=~ INC)
summary(COL.D1.Bayes)
summary(impacts(COL.D1.Bayes, tr=trMatc), short=TRUE, zstats=TRUE)
#data(elect80, package="spData")
#lw <- nb2listw(e80_queen, zero.policy=TRUE)
#el_ml <- lagsarlm(log(pc_turnout) ~ log(pc_college) + log(pc_homeownership) + log(pc_income), data=elect80, listw=lw, zero.policy=TRUE, method="LU")
#summary(el_ml)
#set.seed(1)
#el_B <- spBreg_lag(log(pc_turnout) ~ log(pc_college) + log(pc_homeownership) + log(pc_income), data=elect80, listw=lw, zero.policy=TRUE)
#summary(el_B)
#el_ml$timings
#attr(el_B, " timings")
data(oldcol, package="spdep")
lw <- spdep::nb2listw(COL.nb)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
pslx0 <- predict(COL.SLX)
pslx1 <- predict(COL.SLX, newdata=COL.OLD, listw=lw)
all.equal(pslx0, pslx1)
COL.OLD1 <- COL.OLD
COL.OLD1$INC <- COL.OLD$INC + 1
pslx2 <- predict(COL.SLX, newdata=COL.OLD1, listw=lw)
sum(coef(COL.SLX)[c(2,4)])
mean(pslx2-pslx1)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
summary(COL.SLX)
summary(impacts(COL.SLX))
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL + I(HOVAL^2), data=COL.OLD, listw=lw, Durbin=TRUE)
summary(COL.SLX)
summary(impacts(COL.SLX))
COL.SLX <- lmSLX(CRIME ~ INC, data=COL.OLD, listw=lw)
summary(COL.SLX)
summary(impacts(COL.SLX))
crds <- cbind(COL.OLD$X, COL.OLD$Y)
mdist <- sqrt(sum(diff(apply(crds, 2, range))^2))
dnb <- dnearneigh(crds, 0, mdist)
dists <- nbdists(dnb, crds)
f <- function(x, form, data, dnb, dists, verbose) {
  glst <- lapply(dists, function(d) 1/(d^x))
  lw <- nb2listw(dnb, glist=glst, style="B")
  res <- logLik(lmSLX(form=form, data=data, listw=lw))
  if (verbose) cat("power:", x, " logLik:", res, "\n")
  res
}
opt <- optimize(f, interval=c(0.1, 4), form=CRIME ~ INC + HOVAL, data=COL.OLD, dnb=dnb, dists=dists, verbose=TRUE, maximum=TRUE)
glst <- lapply(dists, function(d) 1/(d^opt$maximum))
lw <- nb2listw(dnb, glist=glst, style="B")
SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
summary(SLX)
summary(impacts(SLX))
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
# fit models for comparison
lm.mod <- lm(CRIME ~ HOVAL + INC, data=columbus)
lag <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
mixed <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb),
Durbin=TRUE)
error <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus, nb2listw(col.gal.nb))
# compare nested models
LR.sarlm(mixed, error)
#anova(lag, lm.mod)
#anova(lag, error, mixed)
AIC(lag, error, mixed)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
error.col <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus,
  nb2listw(col.gal.nb))
bptest.sarlm(error.col)
bptest.sarlm(error.col, studentize=FALSE)
lm.target <- lm(error.col$tary ~ error.col$tarX - 1)
if (require(lmtest) & require(sandwich)) {
  print(coeftest(lm.target, vcov=vcovHC(lm.target, type="HC0"), df=Inf))
}
data(oldcol)
lw <- nb2listw(COL.nb, style="W")
ev <- eigenw(similar.listw(lw))
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, quiet=FALSE, control=list(pre_eig=ev))
summary(COL.errW.eig)
COL.errW.eig_ev <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, control=list(pre_eig=ev))
all.equal(coefficients(COL.errW.eig), coefficients(COL.errW.eig_ev))
COL.errB.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  nb2listw(COL.nb, style="B"))
summary(COL.errB.eig)
W <- as(nb2listw(COL.nb), "CsparseMatrix")
trMatc <- trW(W, type="mult")
COL.errW.M <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, method="Matrix", quiet=FALSE, trs=trMatc)
summary(COL.errW.M)
COL.SDEM.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, etype="emixed", control=list(pre_eig=ev))
summary(COL.SDEM.eig)
COL.SDEM.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  lw, Durbin=TRUE, control=list(pre_eig=ev))
summary(COL.SDEM.eig)
COL.SDEM.eig <- errorsarlm(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
  lw, Durbin=INC, control=list(pre_eig=ev))
summary(COL.SDEM.eig)
summary(impacts(COL.SDEM.eig))
NA.COL.OLD <- COL.OLD
spdep-deprecated

COL.err.NA <- errorsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD, 
nb2listw(COL.nb), na.action=na.exclude)
COL.err.NA$na.action
COL.err.NA
resid(COL.err.NA)
lw <- nb2listw(COL.nb, style="W")
print(system.time(ev <- eigenw(similar.listw(lw))))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="eigen", control=list(pre_eig=ev))))
ocoef <- coefficients(COL.errW.eig)
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="eigen", control=list(pre_eig=ev, LAPACK=FALSE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="Matrix_J", control=list(super=TRUE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="Matrix_J", control=list(super=FALSE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="Matrix", control=list(super=as.logical(NA))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="Matrix", control=list(super=TRUE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="Matrix", control=list(super=FALSE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="Matrix", control=list(super=as.logical(NA))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
if (require("spam", quietly=TRUE)) {
  print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="spam", control=list(spamPivot="MMD")))
  print(all.equal(ocoef, coefficients(COL.errW.eig)))
  print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="spam", control=list(spamPivot="RCM")))
  print(all.equal(ocoef, coefficients(COL.errW.eig)))
  print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="spam_update", control=list(spamPivot="MMD")))
  print(all.equal(ocoef, coefficients(COL.errW.eig)))
  print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="spam_update", control=list(spamPivot="RCM")))
  print(all.equal(ocoef, coefficients(COL.errW.eig)))
  print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
lw, method="spam_update", control=list(spamPivot="RCM")))
  print(all.equal(ocoef, coefficients(COL.errW.eig)))
}
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
listw <- nb2listw(col.gal.nb)
ev <- eigenw(listw)
lobj <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw,
control=list(pre_eig=ev))
summary(lobj)

mobj <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, Durbin=TRUE,
control=list(pre_eig=ev))
summary(mobj)

mobj1 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, Durbin= ~ INC,
control=list(pre_eig=ev))
summary(mobj1)

W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
trMC <- trW(W, type="MC")
set.seed(1)
impacts(lobj, listw=listw)
impacts(lobj, tr=trMatc)
impacts(lobj, tr=trMC)
impacts(lobj, evals=ev)
library(coda)
lobjIQ5 <- impacts(lobj, tr=trMatc, R=200, Q=5)
summary(lobjIQ5, zstats=TRUE, short=TRUE)
summary(lobjIQ5, zstats=TRUE, short=TRUE, reportQ=TRUE)
impacts(mobj, listw=listw)
impacts(mobj, tr=trMatc)
impacts(mobj, tr=trMC)
impacts(mobj1, tr=trMatc)
impacts(mobj1, listw=listw)

set(tr=try(impacts(mobj, evals=ev), silent=TRUE), "\n")
summary(impacts(mobj, tr=trMatc, R=200), short=TRUE, zstats=TRUE)
summary(impacts(mobj1, tr=trMatc, R=200), short=TRUE, zstats=TRUE)
#xobj <- lmSLX(CRIME ~ INC + HOVAL, columbus, listw)
#summary(impacts(xobj))
eobj <- errorsarlm(CRIME ~ INC + HOVAL, columbus, listw, etype="emixed")
summary(impacts(eobj), adjust_k=TRUE)

mobj1 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="Matrix", control=list(fdHess=TRUE))
summary(mobj1)
set.seed(1)
summary(impacts(mobj1, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
summary(impacts(mobj1, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
mobj2 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="Matrix", control=list(fdHess=TRUE, optimHess=TRUE))
summary(impacts(mobj2, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
if (require("spam", quietly=TRUE)) {
  mobj3 <- lagsarlm(CRIME ~ INC + HOVAL, columbus, listw, type="mixed",
method="spam", control=list(fdHess=TRUE))
  summary(impacts(mobj3, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
}
data(boston, package="spData")
Wb <- as(nb2listw(boston.soi), "CsparseMatrix")
trMatb <- trW(Wb, type="mult")
gp2mMi <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) +
I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed",  method="Matrix",
control=list(fdHess=TRUE), tsm=trMatb)
summary(gp2mMi)
summary(impacts(gp2mMi, tr=trMatb, R=1000), zstats=TRUE, short=TRUE)
#data(house, package="spData")
 lw <- nb2listw(L0_nb)
#form <- formula(log(price) ~ age + I(age^2) + I(age^3) + log(lotsize) + 
# rooms + log(TLA) + beds + syear)
 lobj <- lagsarlm(form, house, lw, method="Matrix", 
# control=list(fdHess=TRUE), trs=trMat)
#summary(lobj)
 lobj <- impacts(lobj, tr=trMat, R=1000)
#summary(lobj, zstats=TRUE, short=TRUE)
 lobj1 <- stsls(form, house, lw)
#summary(lobj, zstats=TRUE, short=TRUE)
 mobj <- lagsarlm(form, house, lw, type="mixed", 
# method="Matrix", control=list(fdHess=TRUE), trs=trMat)
#summary(mobj)
 mobj <- impacts(mobj, tr=trMat, R=1000)
#summary(mobj, zstats=TRUE, short=TRUE)

#data(oldcol)
 listw <- nb2listw(COL_nb, style="W")
 ev <- eigenw(listw)
 W <- as(listw, "CsparseMatrix")
 trMatc <- trW(W, type="mult")
 COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw, 
 method="eigen", quiet=FALSE, control=list(pre_eig=ev, OrdVsign=1))
 summary(COL.lag.eig, correlation=TRUE)
 COL.lag.eig$fdHess
 COL.lag.eig$resvar

# using the apparent sign in Ord (1975, equation B.1)
 COL.lag.eigb <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw, 
 method="eigen", control=list(pre_eig=ev, OrdVsign=-1))
 summary(COL.lag.eigb)
 COL.lag.eigb$fdHess
 COL.lag.eigb$resvar

# force numerical Hessian
 COL.lag.eig1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
 listw=listw, method="Matrix", control=list(small=25))
 summary(COL.lag.eig1)
 COL.lag.eig1$fdHess

# force LeSage & Pace (2008, p. 57) approximation
 COL.lag.eig1a <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, 
 listw=listw, method="Matrix", control=list(small=25), trs=trMatc)
 summary(COL.lag.eig1a)
 COL.lag.eig1a$fdHess
 COL.lag.eig1a$resvar[2,2]

# using the apparent sign in Ord (1975, equation B.1)
 COL.lag.eig$fdHess
 COL.lag.eig$resvar[2,2]

# force numerical Hessian
 COL.lag.eig1$fHess[1,1]

# force LeSage & Pace (2008, p. 57) approximation
 COL.lag.eig1$fHess[2,2]

system.time(COL.lag.M <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
spdep-deprecated

nb2listw(COL.nb, method="Matrix", quiet=FALSE))
summary(COL.lag.M)
impacts(COL.lag.M, listw=nb2listw(COL.nb))
if (require("spam", quietly=TRUE)) {
system.time(COL.lag.sp <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb), method="spam", quiet=FALSE))
summary(COL.lag.sp)
}
COL.lag.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="B"))
summary(COL.lag.B)
COL.mixed.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
nb2listw(COL.nb, style="B"), type="mixed", tol.solve=1e-9,
control=list(pre_eig=ev))
summary(COL.mixed.B)
COL.mixed.W <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, type="mixed",
control=list(pre_eig=ev))
summary(COL.mixed.W)
COL.mixed.D00 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, Durbin=TRUE,
control=list(pre_eig=ev))
summary(COL.mixed.D00)
COL.mixed.D01 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, Durbin=FALSE,
control=list(pre_eig=ev))
summary(COL.mixed.D01)
COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, Durbin= ~ INC + HOVAL,
control=list(pre_eig=ev))
summary(COL.mixed.D1)
f <- CRIME ~ INC + HOVAL
COL.mixed.D2 <- lagsarlm(f, data=COL.OLD, listw,
Durbin=as.formula(delete.response(terms(f))),
control=list(pre_eig=ev))
summary(COL.mixed.D2)
COL.mixed.D1a <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, Durbin= ~ INC,
control=list(pre_eig=ev))
summary(COL.mixed.D1a)
try(COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, Durbin= ~ inc + HOVAL,
control=list(pre_eig=ev)))
try(COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
listw, Durbin= ~ DISCBD + HOVAL,
control=list(pre_eig=ev)))
NA.COL.OLD <- COL.OLD
COL.lag.NA <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
nb2listw(COL.nb), na.action=na.exclude,
control=list(tol.opt=.Machine$double.eps*0.4))
COL.lag.NA$na.action
COL.lag.NA
resid(COL.lag.NA)
data(boston, package="spData")
gp2mM <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed", method="Matrix")
summary(gp2mM)

W <- as(nb2listw(boston.soi), "CsparseMatrix")
trMatb <- trW(W, type="mult")
gp2mMi <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2) + AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
data=boston.c, nb2listw(boston.soi), type="mixed", method="Matrix",
trs=trMatb)
summary(gp2mMi)
data(oldcol)
lw <- nb2listw(COL.nb)

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

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

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

sqrt(deviance(COL.mix.eig)/length(COL.nb))
# sqrt(sum((COL.OLD$CRIME - as.vector(p1))^2)/length(COL.nb))
p1 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy.mixed = TRUE)

COL.err.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw)
AIC(COL.err.eig)

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

COL.SDerr.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw,
etype="emixed")
AIC(COL.SDerr.eig)

sqrt(deviance(COL.SDerr.eig)/length(COL.nb))

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

COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw)
AIC(COL.lag.eig)

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

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

# p3 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy=FALSE, legacy.mixed = TRUE)
# all.equal(p2, p3, check.attributes=FALSE)

# p4 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
# legacy=FALSE, power=TRUE, legacy.mixed = TRUE)
data(oldcol)
listw <- nb2listw(COL.nb, style="W")
ev <- eigenw(listw)
COL.sacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw, 
  control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.sacW.eig)
W <- as(listw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
set.seed(1)
summary(impacts(COL.sacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw, 
  type="sacmixed", control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW.eig)
set.seed(1)
summary(impacts(COL.msacW.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
COL.msacW1.eig <- sacsarlm(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD, listw, 
  Durbin=TRUE, control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.msacW1.eig)
set.seed(1)
summary(impacts(COL.msacW1.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
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)
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
listw <- nb2listw(col.gal.nb)
W <- as(listw, "CsparseMatrix")
system.time(trMat <- trW(W, type="mult")
str(trMat)
set.seed(1100)
system.time(trMC <- trW(W, type="MC")
str(trMC)
plot(trMat, trMC)
abline(a=0, b=1)
for(i in 3:length(trMC)) {
  segments(trMat[i], trMC[i]-2*attr(trMC, "sd")[i], trMat[i], 
  trMC[i]+2*attr(trMC, "sd"))[i])
}
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
- \(n1\)  \(n - 1\)
- \(n2\)  \(n - 2\)
- \(n3\)  \(n - 3\)
- \(nn\)  \(n \times n\)
- \(S0\)  global sum of weights
- \(S1\)  \(S1\) sum of weights
- \(S2\)  \(S2\) sum of weights

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

nb2listw

Examples

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

Compute the sum of dissimilarity

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

Elias T. Krainski and Renato M. Assuncao

See Also

See Also as nbcost
Examples

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

Description

The function subsets a spatial weights list, retaining objects for which the subset argument vector is TRUE. At present it will only subset non-general weights lists (that is those created by nb2listw with glist=NULL).

Usage

## S3 method for class 'listw'
subset(x, subset, zero.policy = NULL, ...)

Arguments

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

generic function pass-through

Value

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

nb2listw, subset.nb
Examples

col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[1])
to.be.dropped <- c(31, 34, 36, 39, 42, 46)
pre <- nb2listw(col.gal.nb)
print(pre)
post <- subset(pre, !(1:length(col.gal.nb) %% to.be.dropped))
print(post)

subset.nb

Subset a neighbours list

Description

The function subsets a neighbors list, retaining objects for which the subset argument vector is
TRUE.

Usage

## S3 method for class 'nb'
subset(x, subset, ...)

Arguments

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

```r
columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
coords <- coordinates(as(columbus, "Spatial"))
plot(col.gal.nb, coords)
to.be.dropped <- c(31, 34, 36, 39, 42, 46)
text(coords[to.be.dropped,1], coords[to.be.dropped,2], labels=to.be.dropped,
pos=2, offset=0.3)
sub.col.gal.nb <- subset(col.gal.nb,
  !(1:length(col.gal.nb) %in% to.be.dropped))
plot(sub.col.gal.nb, coords[-to.be.dropped,], col="red", add=TRUE)
which(!(attr(col.gal.nb, "region.id") %in% attr(sub.col.gal.nb, "region.id")))
```

summary.nb

Print and summary function for neighbours and weights lists

Description

The function prints summary measures for links in a neighbours list. If a matrix of coordinates is given as well, summary descriptive measures for the link lengths are also printed. Print and summary functions are also available for "listw" weights list objects, also reporting constants (S0, S1, S2) used in inference for global spatial autocorrelation statistics such as Moran’s I, Geary’s C, join-count tests and Getis-Ord G.

Usage

```r
## S3 method for class 'nb'
summary(object, coords=NULL, longlat = NULL, scale = 1, ...)
## S3 method for class 'nb'
print(x, ...)
## S3 method for class 'listw'
summary(object, coords, longlat, zero.policy = NULL,
  scale = 1, ...)
## S3 method for class 'listw'
print(x, zero.policy = NULL, ...)
```

Arguments

- `object`: an object of class `nb`
- `coords`: matrix of region point coordinates or a SpatialPoints object or an sfc points object
- `longlat`: TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if coords is a SpatialPoints object, the value is taken from the object itself
- `...`: additional arguments affecting the output produced
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 bottom to the top of the screen.

Usage

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

Arguments

- coords: A matrix or a data frame containing the X and Y coordinates of the study sites.
- unit.angle: Character. The measurement units in which angles are defined: either "degrees" (default) or "radians".
- max.dist: Numeric. The maximum distance of an edge linking two sites together.
- tolerance: Numeric. The tolerance angle in which a site can influence another site. The angle is measured vertically and from bottom to top of the pictures after rotation of the points.

Examples

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
coords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
col.gal.nb
summary(col.gal.nb, coords)
col.listw <- nb2listw(col.gal.nb, style="W")
col.listw
summary(col.listw)
tolerance.nb

**rot.angle**

Numeric, optional. An angle at which a set of coordinates should be rotated before creating the connexion diagram. The set of coordinates is rotated counterclockwise. Negative values will produce a clockwise rotation.

**plot.sites**

Logical (TRUE, FALSE) determining if the site should be plotted in a graphic window. This graph allows one to make sure the points are rotated in a correct direction.

**Details**

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

**Value**

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

**Warning**

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

**Author(s)**

F. Guillaume Blanchet

**See Also**

dnearneigh, cell2nb, graphneigh, tri2nb, knn2nb

**Examples**

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

### Construct object of class nb with a tolerance angle of 30 degrees
### and a maximum distance of 2 m.
b.ex<-tolerance.nb(ex.data, unit.angle = "degrees", max.dist=1,
                 tolerance = 30)

### Construct object of class nb with a tolerance angle of 30 degrees
### and a maximum distance of 2 m. The coordinates are rotated at an angle
### of 45 degrees counterclockwise.
b.ex2<-tolerance.nb(ex.data, unit.angle = "degrees", max.dist=1,
                   tolerance = 30, rot.angle = 45)

### Construct object of class nb with a tolerance angle of pi/8 radians
### and a maximum distance of 1.5 m. The coordinates are rotated at
### an angle of pi/4 radians clockwise.
b.ex3<-tolerance.nb(ex.data, unit.angle = "radians", max.dist=1.5,
                   tolerance = pi/8, rot.angle = pi/4)
```

tolerance = pi/8, rot.angle = -pi*2/3

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

---

**tri2nb**

*Neighbours list from tri object*

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

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

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**See Also**

knn2nb, dnearneigh, cell2nb
**Examples**

columbus <- st_read(system.file("shapes/columbus.shp", package="spData")[[1]], quiet=TRUE)
cords <- st_centroid(st_geometry(columbus), of_largest_polygon=TRUE)
ind <- row.names(columbus)
suppressPackageStartupMessages(require(deldir))
col.tri.nb <- tri2nb(cords, row.names=ind)
plot(st_geometry(columbus), border="grey")
plot(col.tri.nb, cords, add=TRUE)
title(main="Raw triangulation links", cex.main=0.6)
x <- seq(0,1,0.1)
y <- seq(0,2,0.2)
try(xy.nb <- tri2nb(xy))
seed <- 1234
xid <- sample(1:nrow(xy))
xy.nb <- tri2nb(xy[xid,])
plot(xy.nb, xy[xid,])

---

**write.nb.gal**

Write a neighbours list as a GAL lattice file

**Description**

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

**Usage**

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

**Arguments**

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

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

**See Also**

- `read.gal`
Examples

col.gal.nb <- read.gal(system.file("weights/columbus.gal", package="spData")[[1]])
GALfile <- tempfile("GAL")
write.nb.gal(col.gal.nb, GALfile)
col.queen <- read.gal(GALfile)
summary(diffnb(col.queen, col.gal.nb))
Index

* cluster
  nbcosts, 111
  plot.skater, 119
  prunecost, 123
  prunemst, 125
  skater, 133
  ssw, 174
* datasets
  columbus, 18
  eire, 30
  oldcol, 115
* data
  bhicv, 13
* graphs
  mstree, 97
  prunecost, 123
* hplot
  plot.mst, 117
  plot.skater, 119
* manip
  Rotation, 129
* multivariate
  ssw, 174
* spatial
  aggregate nb, 5
  airdist, 6
  aple, 7
  aple.mc, 8
  aple.plot, 10
  autocov_dist, 11
  card, 14
  cell2nb, 15
  choynowski, 17
  diffnb, 19
  dnearneigh, 20
  droplinks, 22
  EBest, 24
  EImoran.mc, 25
  E Blocale, 27
  edit nb, 29
  geary, 30
  geary.mc, 31
  geary.test, 33
  globalG.test, 35
  Graph Components, 36
  graphneigh, 38
  grid2nb, 41
  include.self, 42
  invIrM, 43
  is.symmetric.nb, 46
  joincount.mc, 47
  joincount.multi, 48
  joincount.test, 50
  knearneigh, 52
  knn2nb, 54
  lag.listw, 55
  lee, 56
  lee.mc, 57
  lee.test, 59
  listw2sn, 61
  lm.LMtests, 62
  lm.morantest, 64
  lm.morantest.exact, 66
  lm.morantest.sad, 68
  localG, 73
  localmoran, 75
  localmoran.exact, 78
  localmoran.sad, 81
  LOSH, 84
  LOSH.cs, 86
  LOSH.mc, 87
  mat2listw, 89
  moran, 90
  moran.mc, 91
  moran.plot, 93
  moran.test, 95
  mstree, 97
  nb.set.operations, 99
nb2blocknb, 100
nb2INLA, 102
nb2lines, 103
nb2listw, 104
nb2listwdist, 106
nb2mat, 109
nb2WB, 110
nbcosts, 111
nbdist, 112
nbflag, 113
p.adjustSP, 116
plot.nb, 118
poly2nb, 120
probmap, 122
read.gal, 126
read.gwt2nb, 127
set.mcOption, 130
set.spChkOption, 132
sp.correlogram, 137
sp.mantel.mc, 139
spdep, 141
spdep-deprecated, 142
spweights.constants, 172
subset.listw, 175
subset.nb, 176
summary.nb, 177
tolerance.nb, 178
tri2nb, 180
write.nb.gal, 181

* free
plot.mst, 117
prunemst, 125
skater, 133

aggregate.nb, 5
airdist, 6
anova.sarlm (spdep-deprecated), 142
aple, 7, 9, 11
aple.mc, 8, 8, 131
aple.plot, 8, 10
as.data.frame.localmoranex
  (localmoran.exact), 78
as.data.frame.localmoransad
  (localmoran.sad), 81
as.data.frame.sarlm.pred
  (spdep-deprecated), 142
as.spam.listw (spdep-deprecated), 142
as_dgRMatrix_listw (spdep-deprecated), 142
as_dsCMatrix_I (spdep-deprecated), 142
as_dsCMatrix_IrW (spdep-deprecated), 142
as_dsTMatrix_listw (spdep-deprecated), 142
autocov_dist, 11
bbs (columbus), 18
bhicv, 13
boot, 9
bp.test.sarlm (spdep-deprecated), 142
can.be.simmed (spdep-deprecated), 142
card, 14, 16, 21, 39–42, 54, 55, 120, 121
cell2nb, 15, 101, 119, 180
dehb_setup (spdep-deprecated), 142
chkIDs (set.spChkOption), 132
cholesky, 148, 149
choynowski, 17
coeff.gmsar (spdep-deprecated), 142
coeff.lagmess (spdep-deprecated), 142
coeff.sarlm (spdep-deprecated), 142
coeff.spautolm (spdep-deprecated), 142
coeff.sts1s (spdep-deprecated), 142
coerce, listw, CsparseMatrix-method
  (spdep-deprecated), 142
coeff, listw, RsparseMatrix-method
  (spdep-deprecated), 142
coeff, listw, symmetricMatrix-method
  (spdep-deprecated), 142
columbus, 18
col.gal.nb (columbus), 18
COL.nb (oldcol), 115
COL.OLD (oldcol), 115
combo, 18
complement.nb (nb.set.operations), 99
do.dists (columbus), 18
deviation.gmsar (spdep-deprecated), 142
deviation.lagmess (spdep-deprecated), 142
deviation.sarlm (spdep-deprecated), 142
deviance.spaumol (spdep-deprecated), 142
deviance.sts1s (spdep-deprecated), 142
df2sn (nb2lines), 103
diffnb, 19
dist, 112, 124, 134, 174
dnearneigh, 20, 40, 53, 101, 119, 180
do_det (spdep-deprecated), 142

Deprecated, 150
as.data.frame.localmoranex
  (localmoran.exact), 78
as.data.frame.localmoransad
  (localmoran.sad), 81
as.data.frame.sarlm.pred
  (spdep-deprecated), 142
as.spam.listw (spdep-deprecated), 142
as_dgRMatrix_listw (spdep-deprecated), 142

INDEX
dput, 111
droplinks, 22
EBest, 24, 27, 28, 123
EBImoran (EBImoran.mc), 25
EBImoran.mc, 25, 25
EBlocal, 25, 27, 123
eigenw (spdep-deprecated), 142
eigen_setup (spdep-deprecated), 142
errorsarlm, 83
eigenw (spdep-deprecated), 142
fit.gmsar (spdep-deprecated), 142
fit.lagmess (spdep-deprecated), 142
fit.ME_res (spdep-deprecated), 142
frNN, 20
gabrielneigh (graphneigh), 38
geary, 30, 32, 34
geary.mc, 31, 31, 34, 141
geary.test, 31, 32, 33
geos_unary, 38
get.ClusterOption (set.mcOption), 130
gget.coresOption (set.mcOption), 130
gget.listw_is_CsparseMatrix_Option (set.spChkOption), 132
gget.mcOption (set.mcOption), 130
gget.spChkOption (set.spChkOption), 132
gget.VerboseOption (set.spChkOption), 132
gget.ZeroPolicyOption (set.spChkOption), 132
globalG.test, 35
GMargminImage (spdep-deprecated), 142
Gerrorrssar (spdep-deprecated), 142
Graph Components, 36
graph2nb (graphneigh), 38
graphneigh, 38, 179
grid2nb, 41
griffith_sone (spdep-deprecated), 142
HPDinterval.lagImpact (spdep-deprecated), 142
Hausman.test (spdep-deprecated), 142
HPDinterval.lagImpact (spdep-deprecated), 142
impacts (spdep-deprecated), 142
include.self, 42
influence.measures, 94
intersect.nb, 99
intersect.nb (nb.set.operations), 99
intImpacts (spdep-deprecated), 142
invIrM, 43
invIrW (invIrM), 43
is.symmetric.glist (is.symmetric.nb), 46
is.symmetric.nb, 23, 46
Jacobian_W (spdep-deprecated), 142
jacobianSetup (spdep-deprecated), 142
joincount.mc, 47, 51, 141
joincount.multi, 48, 51
joincount.test, 48, 49, 50
kneigh, 21, 40, 52, 55
kNN, 53
knn, 53
knn2nb, 40, 53, 54, 101, 179, 180
l_max (spdep-deprecated), 142
lag.listw, 55
lagmess (spdep-deprecated), 142
lagsarlm (spdep-deprecated), 142
lee, 56, 58, 60
lee.mc, 57, 57, 60
lee.test, 59
lextrB (spdep-deprecated), 142
lextrS (spdep-deprecated), 142
lextrW (spdep-deprecated), 142
listw2lines (nb2lines), 103
listw2mat (nb2mat), 109
listw2sn, 61
listw2star (localmorans.sad), 81
listw2U, 34, 51, 50, 96
listw2U (lm.morantest), 64
listw2WB (nb2WB), 110
lm, 64, 65
lm.LMtests, 62, 65
lm.morantest, 64, 69, 83
lm.morantest.exact, 66, 80
lm.morantest.sad, 67, 68, 83
lmSLX (spdep-deprecated), 142
localAple (aple.plot), 10
localC, 70
localC_perm (localC), 70
localG, 35, 36, 73, 77, 116
localG_perm (localG), 73
localmoran, 75, 83, 94, 116
localmoran.exact, 78
localmoran.sad, 80, 81
localmoran_perm (localmoran), 75
locator, 6
logLik.lagmess (spdep-deprecated), 142
logLik.sarlm (spdep-deprecated), 142
logLik.spautolm (spdep-deprecated), 142
LOSH, 84, 87–89
LOSH.mc, 86, 86
LOSH.mc.s, 86, 86
LR.sarlm (spdep-deprecated), 142
LR1.sarlm (spdep-deprecated), 142
LR1.spautolm (spdep-deprecated), 142
LU_prepermutate_setup
(spdep-deprecated), 142
LU_setup (spdep-deprecated), 142
make.sym.nb (is.symmetric.nb), 46
mat2listw, 89
Matrix_J_setup (spdep-deprecated), 142
Matrix_setup (spdep-deprecated), 142
mcdet_setup (spdep-deprecated), 142
MCMCsample (spdep-deprecated), 142
ME (spdep-deprecated), 142
mom_calc (spdep-deprecated), 142
mom_calc_int2 (spdep-deprecated), 142
moments_setup (spdep-deprecated), 142
moran, 27, 90, 92, 96, 139
moran.mc, 27, 91, 91, 96, 141
moran.plot, 93
moran.test, 91, 92, 95
mstree, 97, 117, 119, 125, 135
n.comp.nb (Graph Components), 36
nb.set.operations, 99
nb2blocknb, 100
nb2INLA, 102
nb2lines, 103
nb2listw, 8, 12, 45, 55, 62, 89, 90, 97, 104, 108, 110, 112, 113, 173, 175, 176
nb2listwdist, 106
nb2mat, 90, 109
nb2WB, 110
nbcost, 174
nbcost (nbcosts), 111
nbcosts, 111
nbdist, 112, 112
nblag, 113, 139
nblag.cumul (nblag), 113
old.make.sym.nb (is.symmetric.nb), 46
oldcol, 115
p.adjust, 63, 116, 139
p.adjustSP, 76, 86, 88, 116
plot.Gabriel (graphneigh), 38
plot.lagImpact (spdep-deprecated), 142
plot.listw (plot.nb), 118
plot.mc.sim (sp.mantel.mc), 139
plot.mst, 117
plot.nb, 29, 37, 118, 178
plot.relative (graphneigh), 38
plot.skater, 119
plot.spcor (sp.correlogram), 137
poly2nb, 42, 101, 112, 120
polys (columbus), 18
powerWeights (spdep-deprecated), 142
ppois, 123
predict.sarlm (spdep-deprecated), 142
predict.SLX (spdep-deprecated), 142
print.gmsar (spdep-deprecated), 142
print.jclist (joincount.test), 50
print.jcmulti (joincount.multi), 48
print.lagImpact (spdep-deprecated), 142
print.lagmess (spdep-deprecated), 142
print.listw (summary.nb), 177
print.LMtestlist (lm.LMtests), 62
print.localmoransad (localmoran.sad), 81
print.lm.morantest.exact (lm.morantest.exact), 66
print.mantel.mc (spdep-deprecated), 142
print.moran.exact (localmoran.exact), 78
print.localmoransad (localmoran.sad), 81
print.mE_res (spdep-deprecated), 142
print.moransad (lm.morantest.exact), 66
print.moransad (lm.morantest.sad), 68
print.nb (summary.nb), 177
print.sarlm (spdep-deprecated), 142
print.SFResult (spdep-deprecated), 142
print.spcor (sp.correlogram), 137
print.sts1s (spdep-deprecated), 142
print.summary.gmsar (spdep-deprecated), 142
print.summary.lagImpact
(spdep-deprecated), 142
print.summary.moransad
(spdep-deprecated), 142
spdep-deprecated, 84, 86, 87–89
spdep-deprecated, 110
spdep-deprecated, 142
spdep-deprecated, 174
spdep-deprecated, 186
spdep-deprecated, 27
spdep-deprecated, 89
spdep-deprecated, 109
spdep-deprecated, 112
spdep-deprecated, 125
spdep-deprecated, 137
spdep-deprecated, 142
spdep-deprecated, 142
spdep-deprecated, 142
spdep-deprecated, 142
spdep-deprecated, 142
spdep-deprecated, 142
INDEX

print.summary.localmoransad
(localmoransad), 81
print.summary.moransad
(lm.morantest.sad), 68
print.summary.sarlm(spdep-deprecated), 142
print.summary.spautolm
(spdep-deprecated), 142
print.summary.stsls(spdep-deprecated), 142
probmap, 17, 25, 28, 122
prunecost, 123
prunemst, 124, 125
read.dat2listw (read.gwt2nb), 127
read.gal, 47, 105, 126, 128, 181
read.geoda (read.gal), 126
read.gwt2nb, 127
relativeneigh (graphneigh), 38
residuals.gmsar (spdep-deprecated), 142
residuals.lagmess (spdep-deprecated), 142
residuals.sarlm (spdep-deprecated), 142
residuals.spautolm (spdep-deprecated), 142
residuals.stsls (spdep-deprecated), 142
Rotation, 129
sacsarlm (spdep-deprecated), 142
SE_classic_setup (spdep-deprecated), 142
SE_interp_setup (spdep-deprecated), 142
SE_whichMin_setup (spdep-deprecated), 142
set.ClusterOption (set.mcOption), 130
set.coresOption (set.mcOption), 130
set.Listw_is_CsparseMatrix_Option
(set.spChkOption), 132
set.mcOption, 130
set.spChkOption, 132
set.VerboseOption (set.spChkOption), 132
set.ZeroPolicyOption (set.spChkOption), 132
setdiff.nb, 99
setdiff.nb (nb.set.operations), 99
similar.listw (nb.set.operations), 99
skater, 117, 119, 133
sn2listw, 104
sn2listw (listw2sn), 61
soi.graph (graphneigh), 38
sp.correlogram, 137
sp.mantel.mc, 31, 139
spam_setup (spdep-deprecated), 142
spam_update_setup (spdep-deprecated), 142
SpatialFiltering (spdep-deprecated), 142
spautolm (spdep-deprecated), 142
spBreg_err (spdep-deprecated), 142
spBreg_lag (spdep-deprecated), 142
spBreg_tac (spdep-deprecated), 142
spdep, 141
spdep-deprecated, 142
spNamedVec (set.spChkOption), 132
spweights.constants, 172
ssw, 174
stsls (spdep-deprecated), 142
subgraph_eigenw (spdep-deprecated), 142
subset.listw, 175
subset.nb, 175, 176
summary.gmsar (spdep-deprecated), 142
summary.lagImpact (spdep-deprecated), 142
summary.lagmess (spdep-deprecated), 142
summary.listw (spdep-deprecated), 142
summary.nb (spdep-deprecated), 177
summary.LMtestlist (lm.LMtests), 62
summary.localmoransad (localmoransad), 81
summary.moransad (lm.morantest.sad), 68
summary.nb, 15, 16, 29, 42, 43, 105, 108, 113, 114, 118, 121, 127, 177
summary.sarlm (spdep-deprecated), 142
summary.spautolm (spdep-deprecated), 142
summary.stsls (spdep-deprecated), 142
sym.attr.nb (is.symmetric.nb), 46
Szero (spweights.constants), 172
tolerance.nb, 178
tri2nb, 101, 179, 180
trw, 149
trW (spdep-deprecated), 142
union.nb, 99
union.nb (nb.set.operations), 99
vcov.sarlm (spdep-deprecated), 142
vi2mrc (cell2snb), 15
Wald1.sarlm (spdep-deprecated), 142
write.nb.gal, 181
write.sn2dat(read.gwt2nb), 127
write.sn2gwt(read.gwt2nb), 127