Package ‘spatialreg’

March 21, 2026

Version 1.4-3

Date 2026-03-20

Title Spatial Regression Analysis

Encoding UTF-8

Depends R (>= 3.3), spData (>= 2.3.1), Matrix, sf

Imports spdep (>= 1.4.1), coda, methods, mvtnorm, boot, splines, LearnBayes, nlme, multcomp, marginaleffects

Suggests parallel, RSpectra, tmap, foreign, spam, knitr, lmtest, expm, sandwich, rmarkdown, igraph, tinytest, codingMatrices

Description A collection of all the estimation functions for spatial cross-sectional models (on lattice/areal data using spatial weights matrices) contained up to now in 'spdep'. These model fitting functions include maximum likelihood methods for cross-sectional models proposed by 'Cliff' and 'Ord' (1973, ISBN:0850860369) and (1981, ISBN:0850860814), fitting methods initially described by 'Ord' (1975) <doi:10.1080/01621459.1975.10480272>. The models are further described by 'Anselin' (1988) <doi:10.1007/978-94-015-7799-1>. Spatial two stage least squares and spatial general method of moment models initially proposed by 'Kelejian' and 'Prucha' (1998) <doi:10.1023/A:1007707430416> and (1999) <doi:10.1111/1468-2354.00027> are provided. Impact methods and MCMC fitting methods proposed by 'LeSage' and 'Pace' (2009) <doi:10.1201/9781420064254> are implemented for the family of cross-sectional spatial regression models. Methods for fitting the log determinant term in maximum likelihood and MCMC fitting are compared by 'Bivand et al.' (2013) <doi:10.1111/gean.12008>, and model fitting methods by 'Bivand' and 'Piras' (2015) <doi:10.18637/jss.v063.i18>; both of these articles include extensive lists of references. A recent review is provided by 'Bivand', 'Millo' and 'Piras' (2021) <doi:10.3390/math9111276>. 'spatialreg' >= 1.1-* corresponded to 'spdep' >= 1.1-1, in which the model fitting functions were deprecated and passed through to 'spatialreg', but masked those in 'spatialreg'. From versions 1.2-*, the functions have been made defunct in 'spdep'. From version 1.3-6, add Anselin-Kelejian (1997) test to `stsls` for residual spatial autocorrelation <doi:10.1177/016001769702000109>.

License GPL-2

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

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

VignetteBuilder knitr

NeedsCompilation yes

RoxygenNote 6.1.1

Author Roger Bivand [cre, aut] (ORCID:
<https://orcid.org/0000-0003-2392-6140>, ROR:
<https://ror.org/04v53s997>),
Gianfranco Piras [aut],
Luc Anselin [ctb] (ORCID: <https://orcid.org/0000-0003-1076-2220>),
Andrew Bernat [ctb],
Eric Blankmeyer [ctb],
Yongwan Chun [ctb] (ORCID: <https://orcid.org/0000-0002-4957-1379>),
Virgilio Gómez-Rubio [ctb] (ORCID:
<https://orcid.org/0000-0002-4791-3072>),
Daniel Griffith [ctb] (ORCID: <https://orcid.org/0000-0001-5125-6450>),
Martin Gubri [ctb] (ORCID: <https://orcid.org/0000-0001-6744-6662>),
Rein Halbersma [ctb],
James LeSage [ctb],
Angela Li [ctb],
Hongfei Li [ctb],
Jielai Ma [ctb],
Abhirup Mallik [ctb, trl],
Giovanni Millo [ctb] (ORCID: <https://orcid.org/0000-0002-0140-6681>),
Kelley Pace [ctb],
Josiah Parry [ctb] (ORCID: <https://orcid.org/0000-0001-9910-865X>),
Pedro Peres-Neto [ctb],
Tobias Rüttenauer [ctb] (ORCID:
<https://orcid.org/0000-0001-5747-9735>),
Mauricio Sarrias [ctb],
JuanTomas Sayago [ctb],
Michael Tiefelsdorf [ctb]

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

Repository CRAN

Date/Publication 2026-03-21 03:50:02 UTC

Contents

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

Description

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

Usage

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

Arguments

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

Details

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

Value

A scalar APLE value.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

nb2listw, aple.mc, aple.plot

Examples

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

References

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

See Also

`aple`, `boot`

Examples

```
## Not run:
wheat <- st_read(system.file("shapes/wheat.gpkg", package="spData")[1], quiet=TRUE)
nbr1 <- spdep::poly2nb(wheat, queen=FALSE)
nbrl <- spdep::nblag(nbr1, 2)
nbr12 <- spdep::nblag_cumul(nbrl)
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$yield_detrend, scale=FALSE)),
 spdep::nb2listw(nbr12, style="W"), nsim=500)
plot(boot_out_ser)
boot_out_ser
library(parallel)
oldCores <- set.coresOption(NULL)
nc <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
# set nc to 1L here
if (nc > 1L) nc <- 1L
invisible(set.coresOption(nc))
set.seed(1L)
if (!get.mcOption()) {
  cl <- makeCluster(nc)
  set.ClusterOption(cl)
} else{
  mc.reset.stream()
}
boot_out_par <- aple.mc(as.vector(scale(wheat$yield_detrend, scale=FALSE)),
    spdep::nb2listw(nbr12, style="W"), nsim=500)
if (!get.mcOption()) {
  set.ClusterOption(NULL)
  stopCluster(cl)
}
boot_out_par
invisible(set.coresOption(oldCores))
RNGkind(oldRNG[1], oldRNG[2])

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Details

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

Value

`aple.plot` returns list with components:

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

`localAple` returns a vector of local APLE values.

Author(s)

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

References

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

See Also

`aple`

Examples

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

## End(Not run)
```

`as.spam.listw` *Spatial neighbour sparse representation*

Description

Interface between Matrix class objects and weights lists. The `as.spam.listw` method converts a `"listw"` object to a sparse matrix as defined in the **spam** package.

Usage

```
as.spam.listw(listw)
listw2U_spam(lw)
listw2U_Matrix(lw)
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))
```

Arguments

`listw, lw`	a `listw` object from for example `nb2listw`
`W`	a `dsTMatrix` object created using `as_dsTMatrix_listw` from a symmetric `listw` object
`rho`	spatial regression coefficient
`n`	length of diagonal for identity matrix
`order`	Power series maximum limit
`X`	A numerical matrix
`tol`	Tolerance for convergence of power series

Author(s)

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

See Also

`nb2listw`

Examples

```
## Not run:
require(sf, quietly=TRUE)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#require(spdep, quietly=TRUE)
col.gal.nb <- spdep::read.gal(system.file("weights/columbus.gal", package="spData")[1])
col.listw <- spdep::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 <- spdep::mat2listw(nyadjmat)
listw_NY <- spdep::nb2listw(nyadjlw$neighbours, style="B")
W_C <- as(listw_NY, "CsparseMatrix")
W_R <- as(listw_NY, "RsparseMatrix")
W_S <- as(listw_NY, "symmetricMatrix")
n <- nrow(W_S)
I <- Diagonal(n)
rho <- 0.1
c(determinant(I - rho * W_S, logarithm=TRUE)$modulus)
sum(log(1 - rho * eigenw(listw_NY)))
nW <- - W_S
nChol <- Cholesky(nW, Imult=8)
n * log(rho) + (2 * c(determinant(update(nChol, nW, 1/rho))$modulus))

## End(Not run)
nb7rt <- spdep::cell2nb(7, 7, torus=TRUE)
x <- matrix(sample(rnorm(500*length(nb7rt))), nrow=length(nb7rt))
```

```
lw <- spdep::nb2listw(nb7rt)
if (FALSE) {
# Only needed in some simulation settings where the input and
# output distributions must agree in all but autocorrelation
e <- 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) residuals(spautolm(v ~ 1, listw=lw,
 method="eigen", control=list(pre_eig=e, fdHess=FALSE))))
x <- apply(x, 2, scale)
}
W <- as(lw, "CsparseMatrix")
system.time(e <- invIrM(nb7rt, rho=0.98, method="solve", feasible=NULL) %*% x)
system.time(ee <- powerWeights(W, rho=0.98, X=x))
str(attr(ee, "internal"))
all.equal(e, as(ee, "matrix"), check.attributes=FALSE)
## Not run:
system.time(ee <- powerWeights(W, rho=0.9, X=x))
system.time(ee <- powerWeights(W, rho=0.98, order=1000, X=x))
all.equal(e, as(ee, "matrix"), check.attributes=FALSE)
nb60rt <- spdep::cell2nb(60, 60, torus=TRUE)
W <- as(spdep::nb2listw(nb60rt), "CsparseMatrix")
set.seed(1)
x <- matrix(rnorm(dim(W)[1]), ncol=1)
system.time(ee <- powerWeights(W, rho=0.3, X=x))
str(as(ee, "matrix"))
obj <- errorsarlm(as(ee, "matrix")[,1] ~ 1, listw=spdep::nb2listw(nb60rt), method="Matrix")
coefficients(obj)

## End(Not run)
```

do_ldet *Spatial regression model Jacobian computations*

Description

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

Usage

```
do_ldet(coef, env, which=1)
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_prepermutate_setup(env, coef=0.1, order=FALSE, which=1)
moments_setup(env, trs=NULL, m, p, type="MC", correct=TRUE, trunc=TRUE, eq7=TRUE, which=1)
SE_classic_setup(env, SE_method="LU", p=16, m=30, nrho=200, interpn=2000,
 interval=c(-1,0.999), SElndet=NULL, which=1)
SE_whichMin_setup(env, SE_method="LU", p=16, m=30, nrho=200, interpn=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)
can.be.simmed(listw)
```

Arguments

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

`order`	default FALSE; used in LU_prepermutate, note warnings given for `lu` method
`trs`	A numeric vector of `m` traces, as from `trW`
`type`	moments trace type, see `trW`
`correct`	default TRUE: use Smirnov correction term, see `trW`
`trunc`	default TRUE: truncate Smirnov correction term, see `trW`
`eq7`	default TRUE; use 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
`interval`	default c(-1,0.999) if interval argument NULL, bounds for SE toolbox emulation grid
`interpn`	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
`listw`	a spatial weights object

Details

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

eigen Classical Ord eigenvalue computations - either:

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

or:

- **pre_eig** pre-computed eigenvalues

and assigns to the environment:

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

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

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

and assigns to the environment:

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

Matrix_J Standard Cholesky decomposition without updating:

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

and assigns to the environment:

- **csrw** sparse spatial weights matrix
- **I** sparse identity matrix
- **super** the value of the `super` argument
- **method** string: “Matrix_J”

spam Standard Cholesky decomposition without updating:

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

and assigns to the environment:

- **csrw** sparse spatial weights matrix
- **I** sparse identity matrix
- **pivot** string — pivot method
- **method** string: “spam”

spam_update Pre-computed Cholesky decomposition with updating:

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

and assigns to the environment:

- **csrw** sparse spatial weights matrix
- **I** sparse identity matrix
- **csrwchol** A Cholesky decomposition for updating
- **method** string: “spam”

LU Standard LU decomposition without updating:

- **listw** A listw spatial weights object
- **n** number of spatial objects

and assigns to the environment:

- **W** sparse spatial weights matrix
- **I** sparse identity matrix
- **method** string: “LU”

LU_prepermutate Standard LU decomposition with updating (pre-computed fill-reducing permutation):

- **listw** A listw spatial weights object
- **n** number of spatial objects

and assigns to the environment:

- **W** sparse spatial weights matrix
- **lu_order** order argument to lu
- **pq** 2-column matrix for row and column permutation for fill-reduction
- **I** sparse identity matrix
- **method** string: “LU”

MC Monte Carlo approximation:

- **listw** A listw spatial weights object

and assigns to the environment:

- **clx** list of Monte Carlo approximation terms (the first two simulated traces are replaced by their analytical equivalents)
- **W** sparse spatial weights matrix
- **method** string: “MC”

cheb Chebyshev approximation:

- **listw** A listw spatial weights object

and assigns to the environment:

- **trT** vector of Chebyshev approximation terms
- **W** sparse spatial weights matrix
- **method** string: “Chebyshev”

moments moments approximation:

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

and assigns to the environment:

- **trs** vector of traces, possibly approximated
- **q12** integer vector of length 2, unit roots terms, ignored until 0.5-52
- **eq7** logical scalar: use equation 7
- **correct** logical scalar: use Smirnov correction term
- **trunc** logical scalar: truncate Smirnov correction term
- **method** string: “moments”

SE_classic :

- **listw** A listw spatial weights object
- **n** number of spatial objects

and assigns to the environment:

- **detval** two column matrix of lndet grid values
- **method** string: “SE_classic”
- **SE_method** string: “LU” or “MC”

SE_whichMin :

- **listw** A listw spatial weights object
- **n** number of spatial objects

and assigns to the environment:

- **detval** two column matrix of lndet grid values
- **method** string: “SE_whichMin”
- **SE_method** string: “LU” or “MC”

SE_interp :

- **listw** A listw spatial weights object
- **n** number of spatial objects

and assigns to the environment:

- **fit** fitted spline object from which to predict lndet values
- **method** string: “SE_interp”
- **SE_method** string: “LU” or “MC”

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

Three set-up functions emulate the behaviour of the Spatial Econometrics toolbox (March 2010) maximum likelihood lndet grid performance. The toolbox lndet functions compute a smaller number of lndet values for a grid of coefficient values (spacing 0.01), and then interpolate to a finer grid of values (spacing 0.001). “SE_classic”, which is an implementation of the SE toolbox code, for example in f_sar.m, appears to have selected a row in the grid matrix one below the correct row when the candidate coefficient value was between 0.005 and 0.01-fuzz, always rounding the row index down. A possible alternative is to choose the index that is closest to the candidate coefficient value (“SE_whichMin”). Another alternative is to fit a spline model to the first stage coarser grid, and pass this fitted model to the log likelihood function to make a point prediction using the candidate coefficient value, rather than finding the grid index (“SE_interp”).

Value

`do_ldet` returns the value of the Jacobian for the calculation method recorded in the environment argument, and for the Monte Carlo approximation, returns a measure of the spread of the approximation as an “sd” attribute; the remaining functions modify the environment in place as a side effect and return nothing.

Author(s)

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

References

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

Bivand, R. S., Hauke, J., and Kossowski, T. (2013). Computing the Jacobian in Gaussian spatial autoregressive models: An illustrated comparison of available methods. *Geographical Analysis*, 45(2), 150-179.

See Also

spautolm, lagsarlm, errorsarlm, Cholesky

Examples

```
data(boston, package="spData")
#require("spdep", quietly=TRUE)
lw <- spdep::nb2listw(boston.soi)
can.sim <- can.be.simmed(lw)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("verbose", FALSE, envir=env)
assign("family", "SAR", envir=env)
eigen_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("verbose", FALSE, envir=env)
assign("family", "SAR", envir=env)
assign("n", length(boston.soi), envir=env)
eigen_pre_setup(env, pre_eig=eigenw(similar.listw(lw)))
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
assign("n", length(boston.soi), envir=env)
Matrix_setup(env, Imult=2, super=FALSE)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("can.sim", can.sim, envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
spam_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
LU_setup(env)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
env <- new.env(parent=globalenv())
assign("listw", lw, envir=env)
assign("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
LU_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("n", length(boston.soi), envir=env)
assign("similar", FALSE, envir=env)
assign("family", "SAR", envir=env)
set.seed(12345)
mcdet_setup(env, p=16, m=30)
get("similar", envir=env)
do_ldet(0.5, env)
rm(env)
```

GMerrorsar *Spatial simultaneous autoregressive error model estimation by GMM*

Description

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

Usage

```
GMerrorsar(formula, data = list(), listw, na.action = na.fail,
 zero.policy = attr(listw, "zero.policy"), method="nlminb", arnoldWied=FALSE,
 control = list(), pars, scaleU=FALSE, verbose=NULL, legacy=FALSE,
 se.lambda=TRUE, returnHcov=FALSE, pWOrder=250, tol.Hcov=1.0e-10)
## S3 method for class 'Gmsar'
summary(object, correlation = FALSE, Hausman=FALSE, ...)
GMargminImage(obj, lambdaseq, s2seq)
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw`	a `listw` object created for example by `nb2listw`
`na.action`	a function (default `na.fail`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `GMerrorsar()` to terminate with an error
`method`	default `"nlminb"`, or optionally a method passed to `optim` to use an alternative optimizer
`arnoldWied`	default FALSE
`control`	A list of control parameters. See details in `optim` or `nlminb`.
`pars`	starting values for λ and σ^2 for GMM optimisation, if missing (default), approximated from initial OLS model as the autocorrelation coefficient corrected for weights style and model sigma squared
`scaleU`	Default FALSE: scale the OLS residuals before computing the moment matrices; only used if the `pars` argument is missing
`verbose`	default NULL, use global option value; if TRUE, reports function values during optimization.
`legacy`	default FALSE - compute using the unfiltered values of the response and right hand side variables; if TRUE - compute the fitted value and residuals from the spatially filtered model using the spatial error parameter
`se.lambda`	default TRUE, use the analytical method described in `http://econweb.umd.edu/~prucha/STATPROG/OLS/desols.pdf`
`returnHcov`	default FALSE, return the Vo matrix for a spatial Hausman test
`tol.Hcov`	the tolerance for computing the Vo matrix (default=1.0e-10)
`pWOrder`	default 250, if returnHcov=TRUE, pass this order to `powerWeights` as the power series maximum limit
`object, obj`	`Gmsar` object from `GMerrorsar`
`correlation`	logical; (default=FALSE), TRUE not available
`Hausman`	if TRUE, the results of the Hausman test for error models are reported
`...`	`summary` arguments passed through
`lambdaseq`	if given, an increasing sequence of lambda values for gridding
`s2seq`	if given, an increasing sequence of sigma squared values for gridding

Details

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

Note that the fitted() function for the output object assumes that the response variable may be reconstructed as the sum of the trend, the signal, and the noise (residuals). Since the values of the response variable are known, their spatial lags are used to calculate signal components (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data.

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

Value

A list object of class `Gmsar`

`type`	"ERROR"
`lambda`	simultaneous autoregressive error coefficient
`coefficients`	GMM coefficient estimates
`rest.se`	GMM coefficient standard errors
`s2`	GMM residual variance
`SSE`	sum of squared GMM errors
`parameters`	number of parameters estimated
`lm.model`	the `lm` object returned when estimating for $\lambda = 0$
`call`	the call used to create this object
`residuals`	GMM residuals
`lm.target`	the `lm` object returned for the GMM fit
`fitted.values`	Difference between residuals and response variable
`formula`	model formula
`aliased`	if not NULL, details of aliased variables
`zero.policy`	zero.policy for this model
`vv`	list of internal bigG and litg components for testing optimisation surface
`optres`	object returned by optimizer
`pars`	start parameter values for optimisation
`Hcov`	Spatial DGP covariance matrix for Hausman test if available
`legacy`	input choice of unfiltered or filtered values
`lambda.se`	value computed if input argument TRUE
`arnoldWied`	were Arnold-Wied moments used
`GMs2`	GM argmin sigma squared
`scaleU`	input choice of scaled OLS residuals
`vcov`	variance-covariance matrix of regression coefficients
`na.action`	(possibly) named vector of excluded or omitted observations if non-default na.action argument used

Author(s)

Luc Anselin and Roger Bivand

References

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

Roger Bivand, Gianfranco Piras (2015). Comparing Implementations of Estimation Methods for Spatial Econometrics. *Journal of Statistical Software*, 63(18), 1-36. doi:10.18637/jss.v063.i18.

See Also

optim, nlminb, errorsarlm

Examples

```
#require("spdep", quietly=TRUE)
data(oldcol, package="spdep")
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="W"), method="eigen")
(x <- summary(COL.errW.eig, Hausman=TRUE))
coef(x)
COL.errW.GM <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="W"), returnHcov=TRUE)
(x <- summary(COL.errW.GM, Hausman=TRUE))
coef(x)
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$s2, pch=3, lwd=2)
contour(aa, levels=signif(levs, 4), add=TRUE)
COL.errW.GM1 <- GMerrorsar(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="W"))
summary(COL.errW.GM1)
require("sf", quietly=TRUE)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[1], quiet=TRUE)
suppressMessages(nyadjmat <- as.matrix(foreign::read.dbf(system.file(
 "misc/nyadjwts.dbf", package="spData")[1])[-1]))
suppressMessages(ID <- as.character(names(foreign::read.dbf(system.file(
 "misc/nyadjwts.dbf", package="spData")[1]))[-1]))
identical(substring(ID, 2, 10), substring(as.character(nydata$AREAKEY), 2, 10))
listw_NY <- spdep::mat2listw(nyadjmat, as.character(nydata$AREAKEY), 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)
```

griffith_sone *Spatial weights matrix eigenvalues*

Description

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

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

where W is the n by n spatial weights matrix, and λ_i are the eigenvalues of W.

Usage

```
eigenw(listw, quiet=NULL)
griffith_sone(P, Q, type="rook")
subgraph_eigenw(nb, glist=NULL, style="W", zero.policy=NULL, quiet=NULL)
```

Arguments

`listw`	a `listw` object created for example by `nb2listw`
`quiet`	default NULL, use global !verbose option value; set to FALSE for short summary
`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"
`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

The `griffith_sone` function function may be used, following Ord and Gasim (for references see Griffith and Sone (1995)), to calculate analytical eigenvalues for binary rook or queen contiguous neighbours where the data are arranged as a regular P times Q grid. The `subgraph_eigenw` function may be used when there are multiple graph components, of which the largest may be handled as a dense matrix. Here the eigenvalues are computed for each subgraph in turn, and catenated to reconstruct the complete set. The functions may be used to provide pre-computed eigenvalues for spatial regression functions.

Value

a numeric or complex vector of eigenvalues of the weights matrix generated from the spatial weights object.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

Cliff, A. D., Ord, J. K. 1981 Spatial processes, Pion, p. 155; Ord, J. K. 1975 Estimation methods for models of spatial interaction, Journal of the American Statistical Association, 70, 120-126.; Griffith, D. A. and Sone, A. (1995). Trade-offs associated with normalizing constant computational simplifications for estimating spatial statistical models. Journal of Statistical Computation and Simulation, 51, 165-183.

See Also

eigen

Examples

```
#require(spdep)
data(oldcol, package="spdep")
W.eig <- eigenw(spdep::nb2listw(COL.nb, style="W"))
1/range(W.eig)
S.eig <- eigenw(spdep::nb2listw(COL.nb, style="S"))
1/range(S.eig)
B.eig <- eigenw(spdep::nb2listw(COL.nb, style="B"))
1/range(B.eig)
# cases for intrinsically asymmetric weights
crds <- cbind(COL.OLD$X, COL.OLD$Y)
k3 <- spdep::knn2nb(spdep::knearneigh(crds, k=3))
spdep::is.symmetric.nb(k3)
k3eig <- eigenw(spdep::nb2listw(k3, style="W"))
is.complex(k3eig)
rho <- 0.5
Jc <- sum(log(1 - rho * k3eig))
# complex eigenvalue Jacobian
Jc
# subgraphs
nc <- attr(k3, "ncomp")
if (is.null(nc)) nc <- spdep::n.comp.nb(k3)
nc$nc
table(nc$comp.id)
k3eigSG <- subgraph_eigenw(k3, style="W")
all.equal(sort(k3eig), k3eigSG)
W <- as(spdep::nb2listw(k3, style="W"), "CsparseMatrix")
I <- diag(length(k3))
Jl <- sum(log(abs(diag(slot(lu(I - rho * W), "U")))))
# LU Jacobian equals complex eigenvalue Jacobian
```

```
Jl
all.equal(Re(Jc), Jl)
# wrong value if only real part used
Jr <- sum(log(1 - rho * Re(k3eig)))
Jr
all.equal(Jr, Jl)
# construction of Jacobian from complex conjugate pairs (Jan Hauke)
Rev <- Re(k3eig)[which(Im(k3eig) == 0)]
# real eigenvalues
Cev <- k3eig[which(Im(k3eig) != 0)]
pCev <- Cev[Im(Cev) > 0]
# separate complex conjugate pairs
RpCev <- Re(pCev)
IpCev <- Im(pCev)
# reassemble Jacobian
Jc1 <- sum(log(1 - rho*Rev)) + sum(log((1 - rho * RpCev)^2 + (rho^2)*(IpCev^2)))
all.equal(Re(Jc), Jc1)
# impact of omitted complex part term in real part only Jacobian
Jc2 <- sum(log(1 - rho*Rev)) + sum(log((1 - rho * RpCev)^2))
all.equal(Jr, Jc2)
# trace of asymmetric (WW) and crossprod of complex eigenvalues for APLE
sum(diag(W %*% W))
crossprod(k3eig)
# analytical regular grid eigenvalues
rg <- spdep::cell2nb(ncol=7, nrow=7, type="rook")
rg_eig <- eigenw(spdep::nb2listw(rg, style="B"))
rg_GS <- griffith_sone(P=7, Q=7, type="rook")
all.equal(rg_eig, rg_GS)
## Not run:
run <- FALSE
if (require("RSpectra", quietly=TRUE)) run <- TRUE
if (run) {
B <- as(spdep::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) {
print(all.equal(range(Re(rg_eig)), c(resn, res1)))
}
if (run) {
lw <- spdep::nb2listw(rg, style="W")
rg_eig <- eigenw(similar.listw(lw))
print(range(Re(rg_eig)))
}
if (run) {
W  <- as(lw, "CsparseMatrix")
print(Re(c(eigs(W, k=1, which="SR")$values, eigs(W, k=1, which="LR")$values)))
}
## End(Not run)
```

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

Description

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

Usage

```
gstsls(formula, data = list(), listw, listw2 = NULL, na.action = na.fail,
   zero.policy = attr(listw, "zero.policy"), pars=NULL, scaleU=FALSE, control = list(),
   verbose=NULL, method="nlminb", robust=FALSE, legacy=FALSE, W2X=TRUE, sig2n_k=FALSE)
## S3 method for class 'Gmsar'
impacts(obj, ..., n = NULL, tr = NULL, R = NULL,
 listw = NULL, evalues=NULL, Q=NULL)
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw`	a `listw` object created for example by `nb2listw`
`listw2`	a `listw` object created for example by `nb2listw`, if not given, set to the same spatial weights as the listw argument
`na.action`	a function (default `na.fail`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `GMerrorsar()` to terminate with an error
`pars`	starting values for λ and σ^2 for GMM optimisation, if missing (default), approximated from initial 2sls model as the autocorrelation coefficient corrected for weights style and model sigma squared
`scaleU`	Default FALSE: scale the OLS residuals before computing the moment matrices; only used if the `pars` argument is missing
`control`	A list of control parameters. See details in optim or nlminb
`verbose`	default NULL, use global option value; if TRUE, reports function values during optimization.

`method`	default nlminb, or optionally a method passed to optim to use an alternative optimizer
`robust`	see `stsls`
`legacy`	see `stsls`
`W2X`	see `stsls`
`sig2n_k`	see `stsls`
`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
`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
`Q`	default NULL, else an integer number of cumulative power series impacts to calculate if `tr` is given
`n`	defaults to `length(obj$residuals)`; in the method for `Gmsar` objects it may be used in panel settings to compute the impacts for cross-sectional weights only, suggested by Angela Parenti

Details

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

Value

A list object of class `Gmsar`

`lambda`	simultaneous autoregressive error coefficient
`coefficients`	GMM coefficient estimates (including the spatial autocorrelation coefficient)
`rest.se`	GMM coefficient standard errors
`s2`	GMM residual variance
`SSE`	sum of squared GMM errors
`parameters`	number of parameters estimated
`lm.model`	NULL
`call`	the call used to create this object
`residuals`	GMM residuals
`lm.target`	NULL

`fitted.values`	Difference between residuals and response variable
`formula`	model formula
`aliased`	NULL
`zero.policy`	zero.policy for this model
`LL`	NULL
`vv`	list of internal bigG and litg components for testing optimisation surface
`optres`	object returned by optimizer
`pars`	start parameter values for optimisation
`Hcov`	NULL
`na.action`	(possibly) named vector of excluded or omitted observations if non-default na.action argument used

Author(s)

Gianfranco Piras and Roger Bivand

References

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

Roger Bivand, Gianfranco Piras (2015). Comparing Implementations of Estimation Methods for Spatial Econometrics. *Journal of Statistical Software*, 63(18), 1-36. doi:10.18637/jss.v063.i18.

See Also

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

Examples

```
#require("spdep", quietly=TRUE)
data(oldcol, package="spdep")
COL.errW.GM <- gstsls(CRIME ~ INC + HOVAL, data=COL.OLD, spdep::nb2listw(COL.nb, style="W"))
summary(COL.errW.GM)
aa <- GMargminImage(COL.errW.GM)
levs <- quantile(aa$z, seq(0, 1, 1/12))
image(aa, breaks=levs, xlab="lambda", ylab="s2")
points(COL.errW.GM$lambda, COL.errW.GM$s2, pch=3, lwd=2)
contour(aa, levels=signif(levs, 4), add=TRUE)
COL.errW.GM <- gstsls(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="W"), scaleU=TRUE)
summary(COL.errW.GM)
listw <- spdep::nb2listw(COL.nb)
W <- as(listw, "CsparseMatrix")
trMat <- trW(W, type="mult")
impacts(COL.errW.GM, tr=trMat)
```

impacts — *Impacts in spatial lag models*

Description

The calculation of impacts for spatial lag and spatial Durbin models is needed in order to interpret the regression coefficients correctly, because of the spillovers between the terms in these data generation processes (unlike the spatial error model). Methods for "SLX" and Bayesian fitted models are also provided, the former do not need MC simulations, while the latter pass through MCMC draws.

Usage

```
#\method{impacts}{sarlm}(obj, \dots, tr, R = NULL, listw = NULL, evalues=NULL,
# useHESS = NULL, Q=NULL)
#\method{impacts}{lagmess}(obj, ..., R=NULL, listw=NULL)
#\method{impacts}{SLX}(obj, ...)
#\method{impacts}{MCMC_sar_g}(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
#\method{impacts}{MCMC_sem_g}(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
#\method{impacts}{MCMC_sac_g}(obj, ..., tr=NULL, listw=NULL, evalues=NULL, Q=NULL)
## 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)
#\method{print}{WXImpact}(x, ...)
#\method{summary}{WXImpact}(object, ..., adjust_k=(attr(object, "type") == "SDEM"))
## S3 method for class 'LagImpact'
HPDinterval(obj, prob = 0.95, ..., choice="direct")
intImpacts(rho, beta, P, n, mu, Sigma, irho, drop2beta, bnames, interval,
 type, tr, R, listw, evalues, tol, empirical, Q, icept, iicept, p, mess=FALSE,
 samples=NULL, zero_fill = NULL, dvars = NULL)
```

Arguments

`obj`	A spatial regression object created by `lagsarlm` 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
`listw`	If `tr` is not given, a spatial weights object as created by `nb2listw`; they must be the same spatial weights as were used in fitting the spatial regression, but do not have to be row-standardised

`evalues`	vector of eigenvalues of spatial weights matrix for impacts calculations
`n`	defaults to `length(obj$residuals)`; in the method for `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
`useHESS`	Use the Hessian approximation (if available) even if the asymptotic coefficient covariance matrix is available; used for comparing methods
`Q`	default NULL, else an integer number of cumulative power series impacts to calculate if `tr` is given
`reportQ`	default NULL; if TRUE and `Q` given as an argument to `impacts`, report impact components
`x`, `object`	LagImpact objects created by `impacts` methods
`zstats`	default FALSE, if TRUE, also return z-values and p-values for the impacts based on the simulations
`short`	default FALSE, if TRUE passed to the print summary method to omit printing of the mcmc summaries
`choice`	One of three impacts: direct, indirect, or total
`trace`	Argument passed to `plot.mcmc`: plot trace plots
`density`	Argument passed to `plot.mcmc`: plot density plots
`prob`	Argument passed to `HPDinterval.mcmc`: a numeric scalar in the interval (0,1) giving the target probability content of the intervals
`adjust_k`	default TRUE if SDEM else FALSE, adjust internal OLS SDEM standard errors by dividing by n rather than (n-k) (default changed and bug fixed after 0.7-8; standard errors now ML in SDEM summary and impacts summary and identical - for SLX use FALSE)
`rho`, `beta`, `P`, `mu`, `Sigma`, `irho`, `drop2beta`, `bnames`, `interval`, `type`, `icept`, `iicept`, `p`, `mess`, `samples`, `zero_fill`, `dvars`	internal arguments shared inside impacts methods
`tol`, `empirical`	deprecated arguments

Details

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

If `R` is given, simulations will be used to create distributions for the impact measures, provided that the fitted model object contains a coefficient covariance matrix. The simulations are made using `mvrnorm` with the coefficients and their covariance matrix from the fitted model.

The simulations are stored as `mcmc` objects as defined in the **coda** package; the objects are used for convenience but are not output by an MCMC process. The simulated values of the coefficients are checked to see that the spatial coefficient remains within its valid interval — draws outside the interval are discarded.

If a model is fitted with the "Durbin=" set to a formula subsetting the explanatory variables, the impacts object returned reports Durbin impacts for variables included in the formula and lag impacts for the other variables.

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

The method for `gmsar` objects is only for those of `type` `SARAR` output by `gstsls`, and assume that the spatial error coefficient is fixed, and thus omitted from the coefficients and covariance matrix used for simulation.

From version 1.4.1, functions for models including spatially lagged independent variables warn on fitting if any of the right-hand side variables are factors. This is because the interpretation of coefficients that are not slopes is unclear when the variable is not interpretable on an unbounded line, such as factors. Factor variable names are shown with the suffix "(F)", others "dy/dx" in output from impact methods. A discussion can be found at `https://github.com/rsbivand/eqc25_talk`.

Value

An object of class LagImpact.

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

`direct`	numeric vector
`indirect`	numeric vector
`total`	numeric vector

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

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

`res`	a list with three components as for the non-simulation case, with a matching `Qres` list attribute if `Q` was given
`sres`	a list with three `mcmc` matrices, for the direct, indirect and total impacts with a matching `Qmcmc` list attribute if `Q` was given

Author(s)

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

References

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

Roger Bivand, Gianfranco Piras (2015). Comparing Implementations of Estimation Methods for Spatial Econometrics. *Journal of Statistical Software*, 63(18), 1-36. doi:10.18637/jss.v063.i18.

See Also

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

Examples

```
require("sf", quietly=TRUE)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#require("spdep", quietly=TRUE)
col.gal.nb <- spdep::read.gal(system.file("weights/columbus.gal", package="spData")[1])
columbus$fEW <- factor(columbus$EW)
columbus$fDISCBD <- ordered(cut(columbus$DISCBD, c(0, 1.5, 3, 4.5, 6)))
run <- require("codingMatrices", quietly=TRUE)
f <- formula(log(CRIME) ~ INC + HOVAL + fDISCBD + fEW)
listw <- spdep::nb2listw(col.gal.nb)
ev <- eigenw(listw)
lobj <- lagsarlm(f, columbus, listw, control=list(pre_eig=ev))
summary(lobj)
if (run) {
contrasts(columbus$fDISCBD) <- "code_diff"
lobjd <- lagsarlm(f, columbus, listw, control=list(pre_eig=ev))
}
mobj <- lagsarlm(f, columbus, listw, Durbin=TRUE, control=list(pre_eig=ev))
summary(mobj)
mobj1 <- lagsarlm(f, columbus, listw, Durbin= ~ INC + HOVAL, 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)
if (run) {
impacts(lobjd, listw=listw)
}
impacts(lobj, tr=trMatc)
impacts(lobj, tr=trMC)
impacts(lobj, evalues=ev)
library(coda)
lobjIQ5 <- impacts(lobj, tr=trMatc, R=200, Q=5)
summary(lobjIQ5, zstats=TRUE, short=TRUE)
summary(lobjIQ5, zstats=TRUE, short=TRUE, reportQ=TRUE)
impacts(mobj, listw=listw)
impacts(mobj, tr=trMatc)
impacts(mobj, tr=trMC)
impacts(mobj1, tr=trMatc)
impacts(mobj1, listw=listw)
## Not run:
try(impacts(mobj, evalues=ev), silent=TRUE)

## End(Not run)
summary(impacts(mobj, tr=trMatc, R=200), short=TRUE, zstats=TRUE)
summary(impacts(mobj1, tr=trMatc, R=200), short=TRUE, zstats=TRUE)
xobj <- lmSLX(f, columbus, listw)
```

```
summary(impacts(xobj))
eobj <- errorsarlm(f, columbus, listw, etype="emixed")
summary(impacts(eobj), adjust_k=TRUE)
## Not run:
mobj1 <- lagsarlm(f, columbus, listw, type="mixed",
method="Matrix", control=list(fdHess=TRUE))
summary(mobj1)
set.seed(1)
summary(impacts(mobj1, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
summary(impacts(mobj, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
mobj2 <- lagsarlm(f, columbus, listw, type="mixed",
method="Matrix", control=list(fdHess=TRUE, optimHess=TRUE))
summary(impacts(mobj2, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)
mobj3 <- lagsarlm(f, columbus, listw, type="mixed",
method="spam", control=list(fdHess=TRUE))
summary(impacts(mobj3, tr=trMatc, R=1000), zstats=TRUE, short=TRUE)

## End(Not run)
## Not run:
data(boston, package="spData")
Wb <- as(spdep::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, spdep::nb2listw(boston.soi), Durbin=TRUE, method="Matrix",
control=list(fdHess=TRUE), trs=trMatb)
summary(gp2mMi)
summary(impacts(gp2mMi, tr=trMatb, R=1000), zstats=TRUE, short=TRUE)
#data(house, package="spData")
#lw <- spdep::nb2listw(LO_nb)
#form <- formula(log(price) ~ age + I(age^2) + I(age^3) + log(lotsize) +
#   rooms + log(TLA) + beds + syear)
#lobj <- lagsarlm(form, house, lw, method="Matrix",
# control=list(fdHess=TRUE), trs=trMat)
#summary(lobj)
#loobj <- impacts(lobj, tr=trMat, R=1000)
#summary(loobj, zstats=TRUE, short=TRUE)
#lobj1 <- stsls(form, house, lw)
#loobj1 <- impacts(lobj1, tr=trMat, R=1000)
#summary(loobj1, zstats=TRUE, short=TRUE)
#mobj <- lagsarlm(form, house, lw, type="mixed",
# method="Matrix", control=list(fdHess=TRUE), trs=trMat)
#summary(mobj)
#moobj <- impacts(mobj, tr=trMat, R=1000)
#summary(moobj, zstats=TRUE, short=TRUE)

## End(Not run)
```

`invIrM` *Compute SAR generating operator*

Description

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

Usage

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

Arguments

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

Details

The `invIrW` function generates the full weights matrix V, checks that rho lies in its feasible range between 1/min(eigen(V)) and 1/max(eigen(V)), and returns the nxn 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 autocorrelection itself, which will mix with the spatial autocorrelection 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 multipled by the power series equivalent of the dense matrix

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

.

Note

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

Author(s)

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

References

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

See Also

`nb2listw`

Examples

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

## End(Not run)
```

`lagmess` *Matrix exponential spatial lag model*

Description

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

Usage

```
lagmess(formula, data = list(), listw, zero.policy = NULL, na.action = na.fail,
 q = 10, start = -2.5, control=list(), method="BFGS", verbose=NULL,
 use_expm=FALSE)
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw`	a `listw` object created for example by `spdep::nb2listw()`
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA - causing `lagmess()` to terminate with an error
`na.action`	a function (default `options("na.action")`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`q`	default 10; number of powers of the spatial weights to use
`start`	starting value for numerical optimization, should be a small negative number
`control`	control parameters passed to `optim`
`method`	default `BFGS`, method passed to `optim`
`verbose`	default NULL, use global option value; if TRUE report function values during optimization
`use_expm`	default FALSE; if TRUE use `expm::expAtv` instead of a truncated power series of W

Details

The underlying spatial lag model:

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

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

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

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

Value

The function returns an object of class `Lagmess` with components:

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

Author(s)

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

References

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

See Also

`lagsarlm`, `optim`

Examples

```
#require(spdep, quietly=TRUE)
data(baltimore, package="spData")
baltimore$AGE <- ifelse(baltimore$AGE < 1, 1, baltimore$AGE)
lw <- spdep::nb2listw(spdep::knn2nb(spdep::knearneigh(cbind(baltimore$X, baltimore$Y), k=7)))
obj1 <- lm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT),
 data=baltimore)
spdep::lm.morantest(obj1, lw)
spdep::lm.LMtests(obj1, lw, test="all")
system.time(obj2 <- lagmess(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw))
(x <- summary(obj2))
coef(x)
has_expm <- require("expm", quietly=TRUE)
if (has_expm) {
system.time(
obj2a <- lagmess(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw, use_expm=TRUE)
)
summary(obj2a)
}
```

```
obj3 <- lagsarlm(log(PRICE) ~ PATIO + log(AGE) + log(SQFT), data=baltimore, listw=lw)
summary(obj3)

data(boston, package="spData")
lw <- spdep::nb2listw(boston.soi)
gp2 <- lagsarlm(log(CMEDV) ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2)
 +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, lw, method="Matrix")
summary(gp2)
gp2a <- lagmess(CMEDV ~ CRIM + ZN + INDUS + CHAS + I(NOX^2) + I(RM^2)
 +  AGE + log(DIS) + log(RAD) + TAX + PTRATIO + B + log(LSTAT),
 data=boston.c, lw)
summary(gp2a)
```

`lextrB` *Find extreme eigenvalues of binary symmetric spatial weights*

Description

The functions find extreme eigenvalues of binary symmetric spatial weights, when these form planar graphs; general weights are not permiited. `l_max` finds the largest eigenvalue using Rayleigh quotient methods of any "listw" object. `lextrB` first calls `l_max`, and uses its output to find the smallest eigenvalue in addition for binary symmetric spatial weights. `lextrW` extends these to find the smallest eigenvalue for intrinsically symmetric row-standardized binary weights matrices (transformed to symmetric through similarity internally). `lextrS` does the same for variance-stabilized ("S" style) intrinsically symmetric binary weights matrices (transformed to symmetric through similarity internally).

Usage

```
lextrB(lw, zero.policy = TRUE, control = list())
lextrW(lw, zero.policy=TRUE, control=list())
lextrS(lw, zero.policy=TRUE, control=list())
l_max(lw, zero.policy=TRUE, control=list())
```

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

Value

The functions return approximations to the extreme eigenvalues with the eigenvectors returned as attributes of this object.

Control arguments

trace report values in while loops, default NULL assuming FALSE; logical

tol tolerance for breaking while loops, default `.Machine$double.eps^(1/2)`; numeric

maxiter maximum number of iterations in while loops, default `6 * (length(lw$neighbours) - 2`; integer

useC use C code, default TRUE, logical (not in `l_max`)

Note

It may be necessary to modify control arguments if warnings about lack of convergence are seen.

Author(s)

Roger Bivand, Yongwan Chun, Daniel Griffith

References

Griffith, D. A. (2004). Extreme eigenfunctions of adjacency matrices for planar graphs employed in spatial analyses. *Linear Algebra and its Applications*, 388:201–219.

Examples

```
data(boston, package="spData")
#require(spdep, quietly=TRUE)
ab.listb <- spdep::nb2listw(boston.soi, style="B")
er <- range(eigenw(ab.listb))
er
res_1 <- lextrB(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 <- spdep::knn2nb(spdep::knearneigh(boston.utm, k=5))
c(l_max(spdep::nb2listw(k5, style="B")))
max(Re(eigenw(spdep::nb2listw(k5, style="B"))))
c(l_max(spdep::nb2listw(k5, style="C")))
max(Re(eigenw(spdep::nb2listw(k5, style="C"))))
ab.listw <- spdep::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
}
## Not run:
ab.listw <- spdep::nb2listw(boston.soi, style="S")
er <- range(eigenw(similar.listw(ab.listw)))
er
res_1 <- lextrS(ab.listw)
c(res_1)

## End(Not run)
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
}
```

`lmSLX` *Spatial Durbin linear (SLX, spatially lagged X) model*

Description

`lmSLX` fits an `lm` model augmented with the spatially lagged RHS variables, including the lagged intercept when the spatial weights are not row-standardised. `create_WX` creates spatially lagged RHS variables, and is exposed for use in model fitting functions.

Usage

```
lmSLX(formula, data = list(), listw, na.action, weights=NULL, Durbin=TRUE,
 zero.policy=NULL, return_impacts=TRUE)
## S3 method for class 'SlX'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
## S3 method for class 'SlX'
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)
## S3 method for class 'summary.SlX'
print(x, digits = max(3L, getOption("digits") - 3L),
 symbolic.cor = x$symbolic.cor, signif.stars = getOption("show.signif.stars"), ...)
## S3 method for class 'SlX'
impacts(obj, ...)
## S3 method for class 'WXimpact'
print(x, ...)
## S3 method for class 'WXimpact'
summary(object, ..., adjust_k=(attr(object, "type") == "SDEM"))
## S3 method for class 'SlX'
predict(object, newdata, listw, zero.policy=NULL, ...)
create_WX(x, listw, zero.policy=NULL, prefix="")
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw`	a `listw` object created for example by `nb2listw`
`na.action`	a function (default `options("na.action")`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the spatial 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.
`weights`	an optional vector of weights to be used in the fitting process. Non-NULL weights can be used to indicate that different observations have different variances (with the values in weights being inversely proportional to the variances); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations (including the case that there are w_i observations equal to y_i and the data have been summarized) - `lm`
`Durbin`	default TRUE for `lmSLX` (Durbin model including WX); if TRUE, full spatial Durbin model; if a formula object, the subset of explanatory variables to lag. From version 1.3-7, the presence of factors (categorical variables) in the Durbin term will give a warning, as it is as yet unknown how spatial lags of categorical variables should be interpreted.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE assign NA
`return_impacts`	default TRUE; may be set FALSE to avoid problems calculating impacts with aliased variables
`digits`	the number of significant digits to use when printing
`correlation`	logical; if `TRUE`, the correlation matrix of the estimated parameters is returned and printed
`symbolic.cor`	logical. If `TRUE`, print the correlations in a symbolic form (see 'symnum') rather than as numbers
`signif.stars`	logical. If `TRUE`, 'significance stars' are printed for each coefficient
`obj`	A spatial regression object created by `lmSLX`
`...`	Arguments passed through
`prefix`	default empty string, may be "lag" in some cases
`x`, `object`	model matrix to be lagged; lagImpact objects created by `impacts` methods
`adjust_k`	default TRUE if SDEM else FALSE, adjust internal OLS SDEM standard errors by dividing by n rather than (n-k) (default changed and bug fixed after 0.7-8; standard errors now ML in SDEM summary and impacts summary and identical - for SLX use FALSE)
`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.

Details

From version 1.4.1, functions for models including spatially lagged independent variables warn on fitting if any of the right-hand side variables are factors. This is because the interpretation of coefficients that are not slopes is unclear when the variable is not interpretable on an unbounded line, such as factors. Factor variable names are shown with the suffix "(F)", others "dy/dx" in output from impact methods. A discussion can be found at https://github.com/rsbivand/eqc25_talk.

Value

The `lmSLX` function returns an "lm" object with a "mixedImps" list of three impact matrixes (impacts and standard errors) for direct, indirect and total impacts; total impacts and their standard errors calculated using `multcomp::glht`.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

See Also

`lm`

Examples

```
data(oldcol, package="spdep")
lw <- spdep::nb2listw(COL.nb, style="W")
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
summary(COL.SLX)
summary(impacts(COL.SLX))
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL + I(HOVAL^2), data=COL.OLD, listw=lw, Durbin=TRUE)
summary(impacts(COL.SLX))
summary(COL.SLX)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL + I(HOVAL^2), data=COL.OLD, listw=lw, Durbin=~INC)
summary(impacts(COL.SLX))
summary(COL.SLX)
COL.SLX <- lmSLX(CRIME ~ INC, data=COL.OLD, listw=lw)
summary(COL.SLX)
summary(impacts(COL.SLX))
## Not run:
crds <- cbind(COL.OLD$X, COL.OLD$Y)
mdist <- sqrt(sum(diff(apply(crds, 2, range))^2))
dnb <- spdep::dnearneigh(crds, 0, mdist)
dists <- spdep::nbdists(dnb, crds)
f <- function(x, form, data, dnb, dists, verbose) {
  glst <- lapply(dists, function(d) 1/(d^x))
  lw <- spdep::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 <- spdep::nb2listw(dnb, glist=glst, style="B")
SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
summary(SLX)
summary(impacts(SLX))

## End(Not run)
COL.SLX <- lmSLX(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw)
pslx0 <- predict(COL.SLX)
pslx1 <- predict(COL.SLX, newdata=COL.OLD, listw=lw)
all.equal(pslx0, pslx1)
COL.OLD1 <- COL.OLD
COL.OLD1$INC <- COL.OLD1$INC + 1
pslx2 <- predict(COL.SLX, newdata=COL.OLD1, listw=lw)
sum(coef(COL.SLX)[c(2,4)])
mean(pslx2-pslx1)
```

LR.Sarlm *Likelihood ratio test*

Description

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

Usage

```
LR.Sarlm(x, y)
## S3 method for class 'Sarlm'
logLik(object, ...)
LR1.Sarlm(object)
Wald1.Sarlm(object)
## S3 method for class 'Sarlm'
Hausman.test(object, ..., tol=NULL)
## S3 method for class 'Sarlm'
anova(object, ...)
bptest.Sarlm(object, varformula=NULL, studentize = TRUE, data=list())
## S3 method for class 'Sarlm'
impacts(obj, ..., tr, R = NULL, listw = NULL, evalues=NULL,
 useHESS = NULL, Q=NULL)
```

Arguments

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

`object, obj`	a `Sarlm` object
`...`	further arguments passed to or from other methods
`tol`	`tol` argument passed to `solve`, default NULL
`varformula`	a formula describing only the potential explanatory variables for the variance (no dependent variable needed). By default the same explanatory variables are taken as in the main regression model
`studentize`	logical. If set to `TRUE` Koenker's studentized version of the test statistic will be used.
`data`	an optional data frame containing the variables in the varformula
`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
`listw`	If `tr` is not given, a spatial weights object as created by `nb2listw`; they must be the same spatial weights as were used in fitting the spatial regression, but do not have to be row-standardised
`evalues`	vector of eigenvalues of spatial weights matrix for impacts calculations
`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
`useHESS`	Use the Hessian approximation (if available) even if the asymptotic coefficient covariance matrix is available; used for comparing methods
`Q`	default NULL, else an integer number of cumulative power series impacts to calculate if `tr` is given

Value

The tests return objects of class `htest` with:

`statistic`	value of statistic
`parameter`	degrees of freedom
`p.value`	Probability value
`estimate`	varies with test
`method`	description of test method

`logLik.Sarlm()` returns an object of class `logLik` `LR1.Sarlm`, `Hausman.Sarlm` and `Wald1.Sarlm` returm objects of class `htest`

Note

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

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>, bptest: Torsten Hothorn and Achim Zeileis, modified by Roger Bivand

References

LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton, pp. 61–63; Pace RK and LeSage J (2008) A spatial Hausman test. *Economics Letters* 101, 282–284. T.S. Breusch & A.R. Pagan (1979), A Simple Test for Heteroscedasticity and Random Coefficient Variation. *Econometrica* **47**, 1287–1294

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

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

See Also

logLik.lm, anova.Sarlm, impacts

Examples

```
require("sf", quietly=TRUE)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#require("spdep", quietly=TRUE)
col.gal.nb <- spdep::read.gal(system.file("weights/columbus.gal", package="spData")[1])
lm.mod <- lm(CRIME ~ HOVAL + INC, data=columbus)
lag <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, spdep::nb2listw(col.gal.nb))
mixed <- lagsarlm(CRIME ~ HOVAL + INC, data=columbus, spdep::nb2listw(col.gal.nb), type="mixed")
error <- errorsarlm(CRIME ~ HOVAL + INC, data=columbus, spdep::nb2listw(col.gal.nb))
Hausman.test(error)
LR.Sarlm(mixed, error)
anova(lag, lm.mod)
anova(lag, error, mixed)
AIC(lag, error, mixed)
bptest.Sarlm(error)
bptest.Sarlm(error, studentize=FALSE)
```

MCMCsamp *MCMC sample from fitted spatial regression*

Description

The MCMCsamp method uses rwmetrop, a random walk Metropolis algorithm, from **LearnBayes** to make MCMC samples from fitted maximum likelihood spatial regression models.

Usage

```
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())
```

Arguments

`object`	A spatial regression model object fitted by maximum likelihood with `spautolm`
`mcmc`	The number of MCMC iterations after burnin
`verbose`	default NULL, use global option value; if TRUE, reports progress
`...`	Arguments passed through
`burnin`	The number of burn-in iterations for the sampler
`scale`	a positive scale parameter
`listw, listw2`	`listw` objects created for example by `nb2listw`; should be the same object(s) used for fitting the model
`control`	list of extra control arguments - see `spautolm`

Value

An object of class “mcmc” suited to **coda**, with attributes: “accept” acceptance rate; “type” input ML fitted model type “SAR”, “CAR”, “SMA”, “lag”, “mixed”, “error”, “sac”, “sacmixed”; “timings” run times

Note

If the acceptance rate is below 0.05, a warning will be issued; consider increasing mcmc.

Author(s)

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

References

Jim Albert (2007) Bayesian Computation with R, Springer, New York, pp. 104-105.

See Also

`rwmetrop`, `spautolm`, `lagsarlm`, `errorsarlm`, `sacsarlm`

Examples

```
require("sf", quietly=TRUE)
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$AREAKEY), 2, 10))
#require("spdep", quietly=TRUE)
listw_NY <- spdep::mat2listw(nyadjmat, as.character(nydata$AREAKEY), style="B")
esar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, family="SAR", method="eigen")
summary(esar1f)
res <- MCMCsamp(esar1f, mcmc=1000, burnin=200, listw=listw_NY)
summary(res)
## Not run:
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)

## End(Not run)
esar0 <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY)
summary(esar0)
res <- MCMCsamp(esar0, mcmc=1000, burnin=200, listw=listw_NY)
summary(res)
## Not run:
esar0w <- errorsarlm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8)
summary(esar0)
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)

## End(Not run)
```

ME *Moran eigenvector GLM filtering*

Description

The Moran eigenvector filtering function is intended to remove spatial autocorrelation from the residuals of generalised linear models. It uses brute force eigenvector selection to reach a subset of such vectors to be added to the RHS of the GLM model to reduce residual autocorrelation to below the specified alpha value. Since eigenvector selection only works on symmetric weights, the weights are made symmetric before the eigenvectors are found (from spdep 0.5-50).

Usage

```
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)
```

Arguments

`formula`	a symbolic description of the model to be fit
`data`	an optional data frame containing the variables in the model
`family`	a description of the error distribution and link function to be used in the model
`weights`	an optional vector of weights to be used in the fitting process

`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting
`na.action`	a function (default `options("na.action")`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the spatial 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.
`listw`	a `listw` object created for example by `nb2listw`
`alpha`	used as a stopping rule to choose all eigenvectors up to and including the one with a p-value exceeding alpha
`nsim`	number of permutations for permutation bootstrap for finding p-values
`verbose`	default NULL, use global option value; if TRUE report eigenvectors selected
`stdev`	if TRUE, p-value calculated from bootstrap permutation standard deviate using `pnorm` with alternative="greater", if FALSE the Hope-type p-value
`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

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

Value

An object of class `Me_res`:

`selection`	a matrix summarising the selection of eigenvectors for inclusion, with columns: **Eigenvector** number of selected eigenvector; **ZI** permutation-based standardized deviate of Moran's I if stdev=TRUE; **pr(ZI)** probability value: if stdev=TRUE of the permutation-based standardized deviate, if FALSE the Hope-type probability value, in both cases on-sided. The first row is the value at the start of the search
`vectors`	a matrix of the selected eigenvectors in order of selection

Author(s)

Roger Bivand and Pedro Peres-Neto

References

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

See Also

`SpatialFiltering`, `glm`

Examples

```
#require("spdep", quietly=TRUE)
data(hopkins, package="spData")
hopkins_part <- hopkins[21:36,36:21]
hopkins_part[which(hopkins_part > 0, arr.ind=TRUE)] <- 1
hopkins.rook.nb <- spdep::cell2nb(16, 16, type="rook")
glmbase <- glm(c(hopkins_part) ~ 1, family="binomial")
lw <- spdep::nb2listw(hopkins.rook.nb, style="B")
set.seed(123)
system.time(MEbinom1 <- ME(c(hopkins_part) ~ 1, family="binomial",
 listw=lw, alpha=0.05, verbose=TRUE, nsim=49))
glmME <- glm(c(hopkins_part) ~ 1 + fitted(MEbinom1), family="binomial")
#anova(glmME, test="Chisq")
coef(summary(glmME))
anova(glmbase, glmME, test="Chisq")
## Not run:
require("sf", quietly=TRUE)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#require("spdep", quietly=TRUE)
col.gal.nb <- spdep::read.gal(system.file("weights/columbus.gal", package="spData")[1])
lw <- spdep::nb2listw(col.gal.nb)
lmbase <- lm(CRIME ~ INC + HOVAL, data=columbus)
lagcol <- SpatialFiltering(CRIME ~ 1, ~ INC + HOVAL, data=columbus,
 nb=col.gal.nb, style="W", alpha=0.1, verbose=TRUE)
lagcol
lmlag <- lm(CRIME ~ INC + HOVAL + fitted(lagcol), data=columbus)
anova(lmbase, lmlag)
set.seed(123)
system.time(lagcol1 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
 listw=lw, alpha=0.1, verbose=TRUE))
lagcol1
lmlag1 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol1), data=columbus)
anova(lmbase, lmlag1)

set.seed(123)
lagcol2 <- ME(CRIME ~ INC + HOVAL, data=columbus, family="gaussian",
 listw=lw, alpha=0.1, stdev=TRUE, verbose=TRUE)
lagcol2
lmlag2 <- lm(CRIME ~ INC + HOVAL + fitted(lagcol2), data=columbus)
anova(lmbase, lmlag2)
NA.columbus <- columbus
NA.columbus$CRIME[20:25] <- NA
```

```
COL.ME.NA <- ME(CRIME ~ INC + HOVAL, data=NA.columbus, family="gaussian",
 listw=lw, alpha=0.1, stdev=TRUE, verbose=TRUE,
 na.action=na.exclude)
COL.ME.NA$na.action
summary(lm(CRIME ~ INC + HOVAL + fitted(COL.ME.NA), data=NA.columbus,
 na.action=na.exclude))
nc.sids <- st_read(system.file("shapes/sids.gpkg", package="spData")[1], quiet=TRUE)
rn <- as.character(nc.sids$FIPS)
ncCC89_nb <- spdep::read.gal(system.file("weights/ncCC89.gal", package="spData")[1],
 region.id=rn)
ncCR85_nb <- spdep::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=spdep::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")

## End(Not run)
```

`ML_models` *Spatial simultaneous autoregressive model estimation by maximum likelihood*

Description

The `lagsarlm` function provides Maximum likelihood estimation of spatial simultaneous autoregressive lag and spatial Durbin (mixed) models of the form:

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

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

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

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

where λ is found by `optimize()` first, and β and other parameters by generalized least squares subsequently. With one of the sparse matrix methods, larger numbers of observations can be handled, but the `interval=` argument may need be set when the weights are not row-standardised. When `etype` is "emixed", a so-called spatial Durbin error model is fitted.

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

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

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

Usage

```
lagsarlm(formula, data = list(), listw, na.action, Durbin, type,
 method="eigen", quiet=NULL, zero.policy=NULL, interval=NULL,
 tol.solve=.Machine$double.eps, trs=NULL, control=list())
errorsarlm(formula, data=list(), listw, na.action, weights=NULL,
 Durbin, etype, method="eigen", quiet=NULL, zero.policy=NULL,
 interval = NULL, tol.solve=.Machine$double.eps, trs=NULL, control=list())
sacsarlm(formula, data = list(), listw, listw2 = NULL, na.action, Durbin, type,
 method="eigen", quiet=NULL, zero.policy=NULL, tol.solve=.Machine$double.eps,
 llprof=NULL, interval1=NULL, interval2=NULL, trs1=NULL, trs2=NULL,
 control = list())
## S3 method for class 'Sarlm'
summary(object, correlation = FALSE, Nagelkerke = FALSE,
 Hausman=FALSE, adj.se=FALSE, ...)
## S3 method for class 'Sarlm'
print(x, ...)
## S3 method for class 'summary.Sarlm'
print(x, digits = max(5, .Options$digits - 3),
 signif.stars = FALSE, ...)
## 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, ...)
## S3 method for class 'Sarlm'
nobs(object, ...)
## S3 method for class 'Sarlm'
set_coef(model, coefs, ...)
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`

`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw, listw2`	a `listw` object created for example by `nb2listw`; if `nb2listw` not given, set to the same spatial weights as the `listw` argument
`na.action`	a function (default `options("na.action")`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`weights`	an optional vector of weights to be used in the fitting process. Non-NULL weights can be used to indicate that different observations have different variances (with the values in weights being inversely proportional to the variances); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations (including the case that there are w_i observations equal to y_i and the data have been summarized) - `lm`
`Durbin`	default FALSE (spatial lag, error or SARAR model); if TRUE, full spatial Durbin model (SDM, SDEM or GNM); if a formula object, the subset of explanatory variables to lag. From version 1.3-7, the presence of factors (categorical variables) in the Durbin term will give a warning, as it is as yet unknown how spatial lags of categorical variables should be interpreted.
`type`	(use the 'Durbin=' argument - retained for backwards compatibility only) default "lag", may be set to "mixed"; when "mixed", the lagged intercept is dropped for spatial weights style "W", that is row-standardised weights, but otherwise included; "Durbin" may be used instead of "mixed"
`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
`method`	"eigen" (default) - the Jacobian is computed as the product of (1 - rho*eigenvalue) using `eigenw`, and "spam" or "Matrix_J" for strictly symmetric weights lists of styles "B" and "C", or made symmetric by similarity (Ord, 1975, Appendix C) if possible for styles "W" and "S", using code from the spam or Matrix packages to calculate the determinant; "Matrix" and "spam_update" provide updating Cholesky decomposition methods; "LU" provides an alternative sparse matrix decomposition approach. In addition, there are "Chebyshev" and Monte Carlo "MC" approximate log-determinant methods; the Smirnov/Anselin (2009) trace approximation is available as "moments". Three methods: "SE_classic", "SE_whichMin", and "SE_interp" are provided experimentally, the first to attempt to emulate the behaviour of Spatial Econometrics toolbox ML fitting functions. All use grids of log determinant values, and the latter two attempt to ameliorate some features of "SE_classic".
`quiet`	default NULL, use !verbose global option value; if FALSE, reports function values during optimization.
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `lagsarlm()` to terminate with an error

`interval`	default is NULL, search interval for autoregressive parameter
`tol.solve`	the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to `solve()` (default=1.0e-10). This may be used if necessary to extract coefficient standard errors (for instance lowering to 1e-12), but errors in `solve()` may constitute indications of poorly scaled variables: if the variables have scales differing much from the autoregressive coefficient, the values in this matrix may be very different in scale, and inverting such a matrix is analytically possible by definition, but numerically unstable; rescaling the RHS variables alleviates this better than setting tol.solve to a very small value
`llprof`	default NULL, can either be an integer, to divide the feasible ranges into a grid of points, or a two-column matrix of spatial coefficient values, at which to evaluate the likelihood function
`trs1, trs2`	default NULL, if given, vectors for each weights object of powered spatial weights matrix traces output by `trW`; when given, used in some Jacobian methods
`interval1, interval2`	default is NULL, search intervals for each weights object for autoregressive parameters
`trs`	default NULL, if given, a vector of powered spatial weights matrix traces output by `trW`; when given, insert the asymptotic analytical values into the numerical Hessian instead of the approximated values; may be used to get around some problems raised when the numerical Hessian is poorly conditioned, generating NaNs in subsequent operations; the use of trs is recommended
`control`	list of extra control arguments - see section below
`object, model`	`Sarlm` object from `lagsarlm`, `errorsarlm` or `sacsarlm`
`coefs`	numerical, replacement coefficients to be inserted into a fitted model
`correlation`	logical; if 'TRUE', the correlation matrix of the estimated parameters including sigma is returned and printed (default=FALSE)
`Nagelkerke`	if TRUE, the Nagelkerke pseudo R-squared is reported
`Hausman`	if TRUE, the results of the Hausman test for error models are reported
`adj.se`	if TRUE, adjust the coefficient standard errors for the number of fitted coefficients
`x`	`Sarlm` object from `lagsarlm`, `errorsarlm` or `sacsarlm` in `print.Sarlm`, summary object from `summary.Sarlm` for `print.summary.Sarlm`
`digits`	the number of significant digits to use when printing
`signif.stars`	logical. If TRUE, "significance stars" are printed for each coefficient.
`...`	further arguments passed to or from other methods

Details

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

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

Note that the fitted() function for the output object assumes that the response variable may be reconstructed as the sum of the trend, the signal, and the noise (residuals). Since the values of the response variable are known, their spatial lags are used to calculate signal components (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data. Refer to the help page of `predict.Sarlm` for discussions and references.

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

From version 1.4.1, functions for models including spatially lagged independent variables warn on fitting if any of the right-hand side variables are factors. This is because the interpretation of coefficients that are not slopes is unclear when the variable is not interpretable on an unbounded line, such as factors. Factor variable names are shown with the suffix "(F)", others "dy/dx" in output from impact methods. A discussion can be found at `https://github.com/rsbivand/eqc25_talk`.

Control arguments

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

returnHcov: (error model) default TRUE, return the Vo matrix for a spatial Hausman test

pWOrder: (error model) default 250, if returnHcov=TRUE and the method is not "eigen", pass this order to `powerWeights` as the power series maximum limit

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

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

optimHessMethod: default "optimHess", may be "nlm" or one of the `optim` methods

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

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

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

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

MC_p: default 16; number of random variates

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

spamPivot: default "MMD", alternative "RCM"

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

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

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

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

SE_method default "LU", may be "MC"

nrho default 200, as in SE toolbox; the size of the first stage lndet grid; it may be reduced to for example 40

interpn default 2000, as in SE toolbox; the size of the second stage lndet grid

small_asy default TRUE; if the method is not "eigen", use asymmetric covariances rather than numerical Hessian ones if n <= small

small default 1500; threshold number of observations for asymmetric covariances when the method is not "eigen"

SElndet default NULL, may be used to pass a pre-computed SE toolbox style matrix of coefficients and their lndet values to the "SE_classic" and "SE_whichMin" methods

LU_order default FALSE; used in "LU_prepermutate", note warnings given for `lu` method

pre_eig default NULL; may be used to pass a pre-computed vector of eigenvalues

return_impacts default TRUE; may be set FALSE to avoid problems calculating impacts with aliased variables

OrdVsign default 1; used to set the sign of the final component to negative if -1 (alpha times ((sigma squared) squared) in Ord (1975) equation B.1).

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

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

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

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

pre_eig1, pre_eig2 default NULL; may be used to pass pre-computed vectors of eigenvalues

Author(s)

Roger Bivand <`Roger.Bivand@nhh.no`>, with thanks to Andrew Bernat for contributions to the asymptotic standard error code.

References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion; Ord, J. K. 1975 Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, 70, 120-126; Anselin, L. 1988 *Spatial econometrics: methods and models.* (Dordrecht: Kluwer); Anselin, L. 1995 SpaceStat, a software program for the analysis of spatial data, version 1.80. Regional Research Institute, West Virginia University, Morgantown, WV; Anselin L, Bera AK (1998) Spatial dependence in linear regression models with an introduction to spatial econometrics. In: Ullah A, Giles DEA (eds) Handbook of applied economic statistics. Marcel Dekker, New York, pp. 237-289; Nagelkerke NJD (1991) A note on a general definition of the coefficient of determination. Biometrika 78: 691-692; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York; LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton.

Roger Bivand, Gianfranco Piras (2015). Comparing Implementations of Estimation Methods for Spatial Econometrics. *Journal of Statistical Software*, 63(18), 1-36. doi:10.18637/jss.v063.i18.

Bivand, R. S., Hauke, J., and Kossowski, T. (2013). Computing the Jacobian in Gaussian spatial autoregressive models: An illustrated comparison of available methods. *Geographical Analysis*, 45(2), 150-179.

See Also

`lm`, `impacts`

Examples

```
data(oldcol, package="spdep")
listw <- spdep::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))
(x <- summary(COL.lag.eig, correlation=TRUE))
coef(x)
## Not run:
COL.lag.eig$fdHess
COL.lag.eig$resvar
# using the apparent sign in Ord (1975, equation B.1)
COL.lag.eigb <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw=listw,
 method="eigen", control=list(pre_eig=ev, OrdVsign=-1))
summary(COL.lag.eigb)
COL.lag.eigb$fdHess
COL.lag.eigb$resvar
# force numerical Hessian
COL.lag.eig1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw=listw, method="Matrix", control=list(small=25))
summary(COL.lag.eig1)
```

```
COL.lag.eig1$fdHess
# force LeSage & Pace (2008, p. 57) approximation
COL.lag.eig1a <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw=listw, method="Matrix", control=list(small=25), trs=trMatc)
summary(COL.lag.eig1a)
COL.lag.eig1a$fdHess
COL.lag.eig$resvar[2,2]
# using the apparent sign in Ord (1975, equation B.1)
COL.lag.eigb$resvar[2,2]
# force numerical Hessian
COL.lag.eig1$fdHess[1,1]
# force LeSage & Pace (2008, p. 57) approximation
COL.lag.eig1a$fdHess[2,2]

## End(Not run)
system.time(COL.lag.M <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, method="Matrix", quiet=FALSE))
summary(COL.lag.M)
impacts(COL.lag.M, listw=listw)
## Not run:
system.time(COL.lag.sp <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw=listw, method="spam", quiet=FALSE))
summary(COL.lag.sp)
COL.lag.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="B"), control=list(pre_eig=ev))
summary(COL.lag.B)
COL.mixed.B <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="B"), type="mixed", tol.solve=1e-9,
 control=list(pre_eig=ev))
summary(COL.mixed.B)
COL.mixed.W <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, type="mixed", control=list(pre_eig=ev))
summary(COL.mixed.W)
COL.mixed.D00 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin=TRUE, control=list(pre_eig=ev))
summary(COL.mixed.D00)
COL.mixed.D01 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin=FALSE, control=list(pre_eig=ev))
summary(COL.mixed.D01)
COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin= ~ INC + HOVAL, control=list(pre_eig=ev))
summary(COL.mixed.D1)
f <- CRIME ~ INC + HOVAL
COL.mixed.D2 <- lagsarlm(f, data=COL.OLD, listw,
 Durbin=as.formula(delete.response(terms(f))),
 control=list(pre_eig=ev))
summary(COL.mixed.D2)
COL.mixed.D1a <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin= ~ INC, control=list(pre_eig=ev))
summary(COL.mixed.D1a)
try(COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin= ~ inc + HOVAL, control=list(pre_eig=ev)))
try(COL.mixed.D1 <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin= ~ DISCBD + HOVAL, control=list(pre_eig=ev)))
NA.COL.OLD <- COL.OLD
NA.COL.OLD$CRIME[20:25] <- NA
COL.lag.NA <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 listw, na.action=na.exclude)
COL.lag.NA$na.action
COL.lag.NA
resid(COL.lag.NA)
COL.lag.NA1 <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 listw, Durbin=~INC) # https://github.com/r-spatial/spatialreg/issues/10
COL.lag.NA1$na.action
COL.lag.NA2 <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 listw, Durbin=~INC, na.action=na.exclude)
COL.lag.NA2$na.action
# https://github.com/r-spatial/spatialreg/issues/11
COL.lag.NA3 <- lagsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 listw, control=list(pre_eig=ev))
COL.lag.NA3$na.action

## End(Not run)

## Not run:
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, spdep::nb2listw(boston.soi), type="mixed", method="Matrix")
summary(gp2mM)
W <- as(spdep::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, spdep::nb2listw(boston.soi), type="mixed", method="Matrix",
trs=trMatb)
summary(gp2mMi)

## End(Not run)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, quiet=FALSE, control=list(pre_eig=ev))
summary(COL.errW.eig)
COL.errW.eig_ev <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, 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,
 spdep::nb2listw(COL.nb, style="B"))
summary(COL.errB.eig)
COL.errW.M <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, method="Matrix", quiet=FALSE, trs=trMatc)
summary(COL.errW.M)
COL.SDEM.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, etype="emixed", control=list(pre_eig=ev))
summary(COL.SDEM.eig)
## Not run:
COL.SDEM.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, Durbin=TRUE, control=list(pre_eig=ev))
summary(COL.SDEM.eig)
COL.SDEM.eig <- errorsarlm(CRIME ~ DISCBD + INC + HOVAL, data=COL.OLD,
 listw, Durbin=~INC, control=list(pre_eig=ev))
summary(COL.SDEM.eig)
summary(impacts(COL.SDEM.eig))
NA.COL.OLD <- COL.OLD
NA.COL.OLD$CRIME[20:25] <- NA
COL.err.NA <- errorsarlm(CRIME ~ INC + HOVAL, data=NA.COL.OLD,
 listw, na.action=na.exclude)
COL.err.NA$na.action
COL.err.NA
resid(COL.err.NA)
print(system.time(ev <- eigenw(similar.listw(listw))))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, 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,
 listw, 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,
 listw, method="eigen", control=list(pre_eig=ev, compiled_sse=TRUE))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, 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,
 listw, 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,
 listw, method="Matrix_J", control=list(super=as.logical(NA)))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
print(system.time(COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, 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,
 listw, 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,
 listw, 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,
 listw, 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,
 listw, 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,
 listw, 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,
 listw, method="spam_update", control=list(spamPivot="RCM"))))
print(all.equal(ocoef, coefficients(COL.errW.eig)))
```

```
## End(Not run)
COL.sacW.eig <- sacsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, listw,
 control=list(pre_eig1=ev, pre_eig2=ev))
summary(COL.sacW.eig)
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 ~ 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)
summary(impacts(COL.msacW2.eig, tr=trMatc, R=2000), zstats=TRUE, short=TRUE)
## Not run:
COL.mix.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, type="mixed", method="eigen")
summary(COL.mix.eig, correlation=TRUE, Nagelkerke=TRUE)
COL.mix.M <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw, type="mixed", method="Matrix")
summary(COL.mix.M, correlation=TRUE, Nagelkerke=TRUE)
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
  spdep::nb2listw(COL.nb, style="W"), method="eigen")
summary(COL.errW.eig, correlation=TRUE, Nagelkerke=TRUE, Hausman=TRUE)

## End(Not run)
```

---

predict.Sarlm — *Prediction for spatial simultaneous autoregressive linear model objects*

Description

predict.Sarlm() calculates predictions as far as is at present possible for for spatial simultaneous autoregressive linear model objects, using Haining's terminology for decomposition into trend, signal, and noise, or other types of predictors — see references.

Usage

```
## S3 method for class 'Sarlm'
predict(object, newdata = NULL, listw = NULL, pred.type = "TS", all.data = FALSE,
 zero.policy = NULL, legacy = TRUE, legacy.mixed = FALSE, power = NULL, order = 250,
 tol = .Machine$double.eps^(3/5), spChk = NULL, ...)
```

```
#\method{predict}{SLX}(object, newdata, listw, zero.policy=NULL, ...)
## S3 method for class 'Sarlm.pred'
print(x, ...)
## S3 method for class 'Sarlm.pred'
as.data.frame(x, ...)
```

Arguments

`object`	`Sarlm` object returned by `lagsarlm`, `errorsarlm` or `sacsarlm`, the method for SLX objects takes the output of `lmSLX`
`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'
`listw`	a `listw` object created for example by `nb2listw`. In the out-of-sample prediction case (ie. if newdata is not NULL), if `legacy.mixed=FALSE` or if `pred.type!="TS"`, it should include both in-sample and out-of-sample spatial units. In this case, if regions of the listw are not in the correct order, they are reordered. See 'Details'
`pred.type`	predictor type — default "TS", use decomposition into trend, signal, and noise ; other types available depending on `newdata`. If `newdata=NULL` (in-sample prediction), "TS", "trend", "TC" and "BP" are available. If `newdata` is not NULL and its row names are the same than the `data` used to fit the model (forecast case), "TS", "trend" and "TC" are available. In other cases (out-of-sample prediction), "TS", "trend", "KP1", "KP2", "KP3", "KP4", "KP5", "TC", "BP", "BPW", "BPN", "TS1", "TC1", "BP1", "BPW1" and "BPN1" are available. See 'Details' and references
`all.data`	(only applies to `pred.type="TC"` and newdata is not NULL) default FALSE: return predictions only for newdata units, if TRUE return predictions for all data units. See 'Details'
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing the function to terminate with an error
`legacy`	(only applies to lag and Durbin (mixed) models for `pred.type="TS"`) default TRUE: use ad-hoc predictor, if FALSE use DGP-based predictor
`legacy.mixed`	(only applies to mixed models if newdata is not NULL) default FALSE: compute lagged variables from both in-sample and out-of-sample units with $[WX]_O$ and $[WX]_S$ where `X=cbind(Xs, Xo)`, if TRUE compute lagged variables independantly between in-sample and out-of-sample units with $W_{OO}X_O$ and $W_{SS}X_S$
`power`	(only applies to lag and Durbin (mixed) models for "TS", "KP1", "KP2", "KP3", "TC", "TC1", "BP", "BP1", "BPN", "BPN1", "BPW" and "BPW1" types) use `powerWeights`, if default NULL, set FALSE if `object$method` is "eigen", otherwise TRUE
`order`	power series maximum limit if `power` is TRUE
`tol`	tolerance for convergence of power series if `power` is TRUE
`spChk`	should the row names of data frames be checked against the spatial objects for identity integrity, TRUE, or FALSE, default NULL to use `get.spChkOption()`

`x`	the object to be printed
`...`	further arguments passed through

Details

The function supports three types of prediction. In-sample prediction is the computation of predictors on the data used to fit the model (`newdata=NULL`). Prevision, also called forecast, is the computation of some predictors (“trend”, in-sample “TC” and out-of-sample “TS”) on the same spatial units than the ones used to fit the model, but with different observations of the variables in the model (row names of `newdata` should have the same row names than the data frame used to fit the model). And out-of-sample prediction is the computation of predictors on other spatial units than the ones used to fit the model (`newdata` has different row names). For extensive definitions, see Goulard et al. (2017).

`pred.type` of predictors are available according to the model of `object` an to the type of prediction. In the two following tables, “yes” means that the predictor can be used with the model, “no” means that `predict.Sarlm()` will stop with an error, and “yes*” means that the predictor is not designed for the specified model, but it can be used with `predict.Sarlm()`. In the last case, be careful with the computation of a inappropriate predictor.

In-sample predictors by models

pred.type	sem (mixed)	lag (mixed)	sac (mixed)
“trend”	yes	yes	yes
“TS”	yes	yes	no
“TC”	no	yes	yes*
“BP”	no	yes	yes*

Note that only “trend” and “TC” are available for prevision.

Out-of-sample predictors by models

pred.type	sem (mixed)	lag (mixed)	sac (mixed)
“trend”	yes	yes	yes
“TS”	yes	yes	no
“TS1” or “KP4”	no	yes	yes
“TC”	no	yes	yes*
“TC1” or “KP1”	yes	yes	yes
“BP”	no	yes	yes*
“BP1”	no	yes	yes*
“BPW”	no	yes	yes*
“BPW1”	no	yes	yes*
“BN”	no	yes	yes*
“BPN1”	no	yes	yes*
“KP2”	yes	yes	yes
“KP3”	yes	yes	yes
“KP5”	yes	no	yes*

Values for `pred.type=` include “TS1”, “TC”, “TC1”, “BP”, “BP1”, “BPW”, “BPW1”, “BPN”, “BPN1”, following the notation in Goulard et al. (2017), and for `pred.type=` “KP1”, “KP2”, “KP3”, “KP4”, “KP5”, following the notation in Kelejian et al. (2007). `pred.type="TS"` is described bellow and in Bivand (2002).

In the following, the trend is the non-spatial smooth, the signal is the spatial smooth, and the noise is the residual. The fit returned by `pred.type="TS"` is the sum of the trend and the signal.

When `pred.type="TS"`, the function approaches prediction first by dividing invocations between those with or without newdata. When no newdata is present, the response variable may be reconstructed as the sum of the trend, the signal, and the noise (residuals). Since the values of the response variable are known, their spatial lags are used to calculate signal components (Cressie 1993, p. 564). For the error model, trend = $X\beta$, and signal = $\lambda W y - \lambda W X \beta$. For the lag and mixed models, trend = $X\beta$, and signal = $\rho W y$.

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

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

In the lag model, the signal can be expressed in the following way (for legacy=TRUE):

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

giving a feasible signal component of:

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

For legacy=FALSE, the trend is computed first as:

$$X\beta$$

next the prediction using the DGP:

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

and the signal is found as the difference between prediction and trend. The numerical results for the legacy and DGP methods are identical.

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

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

If `newdata` has more than one row, leave-one-out predictors (`pred.type=` include “TS1”, “TC1”, “BP1”, “BPW1”, “BPN1”, “KP1”, “KP2”, “KP3”, “KP4”, “KP5”) are computed separatly on each out-of-sample unit.

`listw` should be provided except if `newdata=NULL` and `pred.type=` include “TS”, “trend”, or if `newdata` is not `NULL`, `pred.type="trend"` and `object` is not a mixed model.

`all.data` is useful when some out-of-sample predictors return different predictions for in-sample units, than the same predictor type computed only on in-sample data.

Value

`predict.Sarlm()` returns a vector of predictions with three attribute vectors of trend, signal (only for `pred.type="TS"`) and region.id values and two other attributes of pred.type and call with class `Sarlm.pred`.

`print.Sarlm.pred()` is a print function for this class, printing and returning a data frame with columns: "fit", "trend" and "signal" (when available) and with region.id as row names.

Author(s)

Roger Bivand <`Roger.Bivand@nhh.no`> and Martin Gubri

References

Haining, R. 1990 *Spatial data analysis in the social and environmental sciences*, Cambridge: Cambridge University Press, p. 258; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York; Michel Goulard, Thibault Laurent & Christine Thomas-Agnan, 2017 *About predictions in spatial autoregressive models: optimal and almost optimal strategies*, Spatial Economic Analysis Volume 12, Issue 2–3, 304–325 doi:10.1080/17421772.2017.1300679, ; Kelejian, H. H. and Prucha, I. R. 2007 *The relative efficiencies of various predictors in spatial econometric models containing spatial lags*, Regional Science and Urban Economics, Volume 37, Issue 3, 363–374; Bivand, R. 2002 *Spatial econometrics functions in R: Classes and methods*, Journal of Geographical Systems, Volume 4, No. 4, 405–421

See Also

`errorsarlm`, `lagsarlm`, `sacsarlm`

Examples

```
data(oldcol, package="spdep")
lw <- spdep::nb2listw(COL.nb)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw)

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

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

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

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

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

p3 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
 legacy=FALSE, legacy.mixed = TRUE)
all.equal(p2, p3, check.attributes=FALSE)
p4 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
 legacy=FALSE, power=TRUE, legacy.mixed = TRUE)
all.equal(p2, p4, check.attributes=FALSE)
p5 <- predict(COL.mix.eig, newdata=COL.OLD, listw=lw, pred.type = "TS",
 legacy=TRUE, power=TRUE, legacy.mixed = TRUE)
all.equal(p2, p5, check.attributes=FALSE)
```

set.mcOption *Options for parallel support*

Description

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

Usage

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

Arguments

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

Details

Options in the spatialreg 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 mom_calc, including treatment of seeding of RNG for multicore/cluster.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

Examples

```
ls(envir=spatialreg:::.spatialregOptions)
library(parallel)
nc <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
nc
# set nc to 1L here
if (nc > 1L) nc <- 1L
#nc <- ifelse(nc > 2L, 2L, nc)
coresOpt <- get.coresOption()
coresOpt
if (!is.na(nc)) {
 invisible(set.coresOption(nc))
 print(exists("mom_calc"))
 if(.Platform$OS.type == "windows") {
# forking not permitted on Windows - start cluster
# removed for Github actions 210502
## Not run:
  print(get.mcOption())
  cl <- makeCluster(get.coresOption())
  print(clusterEvalQ(cl, exists("mom_calc")))
  set.ClusterOption(cl)
  clusterEvalQ(get.ClusterOption(), library(spatialreg))
  print(clusterEvalQ(cl, exists("mom_calc")))
  clusterEvalQ(get.ClusterOption(), detach(package:spatialreg))
  set.ClusterOption(NULL)
  print(clusterEvalQ(cl, exists("mom_calc")))
  stopCluster(cl)

## End(Not run)
 } else {
  mcOpt <- get.mcOption()
  print(mcOpt)
  print(mclapply(1:get.coresOption(), function(i) exists("mom_calc"),
   mc.cores=get.coresOption()))
  invisible(set.mcOption(FALSE))
  cl <- makeCluster(nc)
  print(clusterEvalQ(cl, exists("mom_calc")))
  set.ClusterOption(cl)
  clusterEvalQ(get.ClusterOption(), library(spatialreg))
  print(clusterEvalQ(cl, exists("mom_calc")))
  clusterEvalQ(get.ClusterOption(), detach(package:spatialreg))
  set.ClusterOption(NULL)
  print(clusterEvalQ(cl, exists("mom_calc")))
  stopCluster(cl)
  invisible(set.mcOption(mcOpt))
 }
 invisible(set.coresOption(coresOpt))
}
```

---

set.ZeroPolicyOption *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.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

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.

Author(s)

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

Examples

```
get.VerboseOption()
get.ZeroPolicyOption()
```

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

Description

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

Usage

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

Arguments

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

Value

a `listw` object

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

lagsarlm, errorsarlm

Examples

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

SpatialFiltering *Semi-parametric spatial filtering*

Description

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

Usage

```
SpatialFiltering(formula, lagformula=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)
```

Arguments

`formula`	a symbolic description of the model to be fit, assuming a spatial error representation; when lagformula is given, it should include only the response and the intercept term
`lagformula`	An extra one-sided formula to be used when a spatial lag representation is desired; the intercept is excluded within the function if present because it is part of the formula argument, but excluding it explicitly in the lagformula argument in the presence of factors generates a collinear model matrix
`data`	an optional data frame containing the variables in the model
`nb`	an object of class `nb`
`glist`	list of general weights corresponding to neighbours
`style`	`style` can take values W, B, C, U, and S
`na.action`	a function (default `options("na.action")`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the spatial weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`zero.policy`	default NULL, use global option value; if FALSE stop with error for any empty neighbour sets, if TRUE permit the weights list to be formed with zero-length weights vectors
`tol`	tolerance value for convergence of spatial filtering
`zerovalue`	eigenvectors with eigenvalues of an absolute value smaller than zerovalue will be excluded in eigenvector search
`ExactEV`	Set ExactEV=TRUE to use exact expectations and variances rather than the expectation and variance of Moran's I from the previous iteration, default FALSE
`symmetric`	Should the spatial weights matrix be forced to symmetry, default TRUE
`alpha`	if not NULL, used instead of the tol= argument as a stopping rule to choose all eigenvectors up to and including the one with a probability value exceeding alpha.
`alternative`	a character string specifying the alternative hypothesis, must be one of greater, less or two.sided (default).
`verbose`	default NULL, use global option value; if TRUE report eigenvectors selected

Value

An `SfResult` object, with:

`selection`	a matrix summarising the selection of eigenvectors for inclusion, with columns: **Step** Step counter of the selection procedure **SelEvec** number of selected eigenvector (sorted descending) **Eval** its associated eigenvalue **MinMi** value Moran's I for residual autocorrelation

ZMinMi standardized value of Moran's I assuming a normal approximation

pr(ZI) probability value of the permutation-based standardized deviate for the given value of the alternative argument

R2 R^2 of the model including exogenous variables and eigenvectors

gamma regression coefficient of selected eigenvector in fit

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

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

Author(s)

Yongwan Chun, Michael Tiefelsdorf, Roger Bivand

References

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

See Also

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

Examples

```
require("sf", quietly=TRUE)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#require("spdep", quietly=TRUE)
col.gal.nb <- spdep::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)
(x <- summary(lmsar))
coef(x)
anova(lmbase, lmsar)
spdep::lm.morantest(lmsar, spdep::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)
spdep::lm.morantest(lmlag, spdep::nb2listw(col.gal.nb))
NA.columbus <- columbus
NA.columbus$CRIME[20:25] <- NA
COL.SF.NA <- SpatialFiltering(CRIME ~ INC + HOVAL, data=NA.columbus,
 nb=col.gal.nb, style="W", na.action=na.exclude)
COL.SF.NA$na.action
summary(lm(CRIME ~ INC + HOVAL + fitted(COL.SF.NA), data=NA.columbus,
 na.action=na.exclude))
```

spautolm — *Spatial conditional and simultaneous autoregression model estimation*

Description

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

Usage

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

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw`	a `listw` object created for example by `nb2listw`
`weights`	an optional vector of weights to be used in the fitting process
`na.action`	a function (default `options("na.action")`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`family`	character string: either `"SAR"` or `"CAR"` for simultaneous or conditional autoregressions; `"SMA"` for spatial moving average added thanks to Jielai Ma - `"SMA"` is only implemented for method=`"eigen"` because it necessarily involves dense matrices
`method`	character string: default `"eigen"` for use of dense matrices, `"Matrix_J"` for sparse matrices (restricted to spatial weights symmetric or similar to symmetric) using methods in the Matrix package; “Matrix” provides updating Cholesky decomposition methods. Values of method may also include "LU", which provides an alternative sparse matrix decomposition approach, and the "Chebyshev" and Monte Carlo "MC" approximate log-determinant methods.

`verbose`	default NULL, use global option value; if TRUE, reports function values during optimization.
`trs`	default NULL, if given, a vector of powered spatial weights matrix traces output by `trW`; when given, used in some Jacobian methods
`interval`	search interval for autoregressive parameter when not using method="eigen"; default is c(-1,0.999), `optimize` will reset NA/NaN to a bound and gives a warning when the interval is poorly set; method="Matrix" will attempt to search for an appropriate interval, if find_interval=TRUE (fails on some platforms)
`zero.policy`	default NULL, use global option value; Include list of no-neighbour observations in output if TRUE — otherwise zero.policy is handled within the listw argument
`tol.solve`	the tolerance for detecting linear dependencies in the columns of matrices to be inverted - passed to `solve()` (default=double precision machine tolerance). Errors in `solve()` may constitute indications of poorly scaled variables: if the variables have scales differing much from the autoregressive coefficient, the values in this matrix may be very different in scale, and inverting such a matrix is analytically possible by definition, but numerically unstable; rescaling the RHS variables alleviates this better than setting tol.solve to a very small value
`llprof`	default NULL, can either be an integer, to divide the feasible range into llprof points, or a sequence of spatial coefficient values, at which to evaluate the likelihood function
`control`	list of extra control arguments - see section below
`object`	`Spautolm` object from `spautolm`
`correlation`	logical; if 'TRUE', the correlation matrix of the estimated parameters is returned and printed (default=FALSE)
`adj.se`	if TRUE, adjust the coefficient standard errors for the number of fitted coefficients
`Nagelkerke`	if TRUE, the Nagelkerke pseudo R-squared is reported
`...`	further arguments passed to or from other methods

Details

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

Note that the fitted() function for the output object assumes that the response variable may be reconstructed as the sum of the trend, the signal, and the noise (residuals). Since the values of the response variable are known, their spatial lags are used to calculate signal components (Cressie 1993, p. 564). This differs from other software, including GeoDa, which does not use knowledge of the response variable in making predictions for the fitting data.

Value

A list object of class `Spautolm`:

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

Control arguments

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

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

- **optimHess:** default FALSE, use `fdHess` from **nlme**, if TRUE, use `optim` to calculate Hessian at optimum
- **optimHessMethod:** default "optimHess", may be "nlm" or one of the `optim` methods
- **Imult:** default 2; used for preparing the Cholesky decompositions for updating in the Jacobian function
- **super:** if NULL (default), set to FALSE to use a simplicial decomposition for the sparse Cholesky decomposition and method "Matrix_J", set to `as.logical(NA)` for method "Matrix", if TRUE, use a supernodal decomposition
- **cheb_q:** default 5; highest power of the approximating polynomial for the Chebyshev approximation
- **MC_p:** default 16; number of random variates
- **MC_m:** default 30; number of products of random variates matrix and spatial weights matrix
- **type** default "MC", used with method "moments"; alternatives "mult" and "moments", for use if `trs` is missing, `trW`
- **correct** default TRUE, used with method "moments" to compute the Smirnov/Anselin correction term
- **trunc** default TRUE, used with method "moments" to truncate the Smirnov/Anselin correction term
- **SE_method** default "LU", may be "MC"
- **nrho** default 200, as in SE toolbox; the size of the first stage lndet grid; it may be reduced to for example 40
- **interpn** default 2000, as in SE toolbox; the size of the second stage lndet grid
- **small_asy** default TRUE; if the method is not "eigen", use asymmetric covariances rather than numerical Hessian ones if n <= small
- **small** default 1500; threshold number of observations for asymmetric covariances when the method is not "eigen"
- **SElndet** default NULL, may be used to pass a pre-computed SE toolbox style matrix of coefficients and their lndet values to the "SE_classic" and "SE_whichMin" methods
- **LU_order** default FALSE; used in "LU_prepermutate", note warnings given for `lu` method
- **pre_eig** default NULL; may be used to pass a pre-computed vector of eigenvalues

Note

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

Author(s)

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

References

Cliff, A. D., Ord, J. K. 1981 *Spatial processes*, Pion; Ord, J. K. 1975 Estimation methods for models of spatial interaction, *Journal of the American Statistical Association*, 70, 120-126; Waller, L. A., Gotway, C. A. 2004 *Applied spatial statistics for public health*, Wiley, Hoboken, NJ, 325-380; Cressie, N. A. C. 1993 *Statistics for spatial data*, Wiley, New York, 548-568; Ripley, B. D. 1981 *Spatial statistics*, Wiley, New York, 88-95; LeSage J and RK Pace (2009) Introduction to Spatial Econometrics. CRC Press, Boca Raton.

See Also

`optimize`, `errorsarlm`, `do_ldet`

Examples

```
require("sf", quietly=TRUE)
nydata <- st_read(system.file("shapes/NY8_bna_utm18.gpkg", package="spData")[1], quiet=TRUE)
## Not run:
lm0 <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata)
summary(lm0)
lm0w <- lm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata, weights=POP8)
summary(lm0w)

## End(Not run)
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$AREAKEY), 2, 10))
#require("spdep", quietly=TRUE)
listw_NY <- spdep::mat2listw(nyadjmat, as.character(nydata$AREAKEY), style="B")
eigs <- eigenw(listw_NY)
## Not run:
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",
 control=list(pre_eig=eigs)))
res <- summary(esar1f)
print(res)
coef(res)
sqrt(diag(res$resvar))
sqrt(diag(esar1f$fit$imat)*esar1f$fit$s2)
sqrt(diag(esar1f$fdHess))
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="SAR", method="Matrix"))
summary(esar1M)
system.time(esar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="SAR", method="Matrix",
 control=list(super=TRUE)))
summary(esar1M)
```

```
esar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="eigen",
 control=list(pre_eig=eigs))
summary(esar1wf)
system.time(esar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, weights=POP8, family="SAR", method="Matrix"))
summary(esar1wM)
esar1wlu <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="LU")
summary(esar1wlu)
esar1wch <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="SAR", method="Chebyshev")
summary(esar1wch)

## End(Not run)
ecar1f <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, family="CAR", method="eigen",
 control=list(pre_eig=eigs))
summary(ecar1f)
## Not run:
system.time(ecar1M <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, family="CAR", method="Matrix"))
summary(ecar1M)

## End(Not run)
ecar1wf <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME, data=nydata,
 listw=listw_NY, weights=POP8, family="CAR", method="eigen",
 control=list(pre_eig=eigs))
summary(ecar1wf)
## Not run:
system.time(ecar1wM <- spautolm(Z ~ PEXPOSURE + PCTAGE65P + PCTOWNHOME,
 data=nydata, listw=listw_NY, weights=POP8, family="CAR", method="Matrix"))
summary(ecar1wM)

## End(Not run)
## Not run:
require("sf", quietly=TRUE)
nc.sids <- st_read(system.file("shapes/sids.gpkg", 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 <- spdep::dnearneigh(cbind(nc.sids$east, nc.sids$north), 0, 30,
 row.names=row.names(nc.sids))
sids.nhbr30.dist <- spdep::nbdists(sids.nhbr30, cbind(nc.sids$east, nc.sids$north))
sids.nhbr <- spdep::listw2sn(spdep::nb2listw(sids.nhbr30,
 glist=sids.nhbr30.dist, style="B", zero.policy=TRUE))
dij <- sids.nhbr[,3]
n <- nc.sids$BIR74
el1 <- min(dij)/dij
el2 <- sqrt(n[sids.nhbr$to]/n[sids.nhbr$from])
sids.nhbr$weights <- el1*el2
sids.nhbr.listw <- spdep::sn2listw(sids.nhbr)
both <- factor(paste(nc.sids$L_id, nc.sids$M_id, sep=":"))
```

```
ft.NWBIR74 <- sqrt(1000)*(sqrt(nc.sids$NWBIR74/nc.sids$BIR74) +
 sqrt((nc.sids$NWBIR74+1)/nc.sids$BIR74))
mdata <- data.frame(both, ft.NWBIR74, ft.SID74, BIR74=nc.sids$BIR74)
outl <- which.max(rstandard(lm_nc))
as.character(nc.sids$NAME[outl])
mdata.4 <- mdata[-outl,]
W <- spdep::listw2mat(sids.nhbr.listw)
W.4 <- W[-outl, -outl]
sids.nhbr.listw.4 <- spdep::mat2listw(W.4)
esarI <- errorsarlm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
 zero.policy=TRUE)
summary(esarI)
esarIa <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
 family="SAR")
summary(esarIa)
esarIV <- errorsarlm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbr.listw,
 zero.policy=TRUE)
summary(esarIV)
esarIVa <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata, listw=sids.nhbr.listw,
 family="SAR")
summary(esarIVa)
esarIaw <- spautolm(ft.SID74 ~ 1, data=mdata, listw=sids.nhbr.listw,
 weights=BIR74, family="SAR")
summary(esarIaw)
esarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata, listw=sids.nhbr.listw,
 weights=BIR74, family="SAR")
summary(esarIIaw)
esarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata,
 listw=sids.nhbr.listw, weights=BIR74, family="SAR")
summary(esarIVaw)
ecarIaw <- spautolm(ft.SID74 ~ 1, data=mdata.4, listw=sids.nhbr.listw.4,
 weights=BIR74, family="CAR")
summary(ecarIaw)
ecarIIaw <- spautolm(ft.SID74 ~ both - 1, data=mdata.4,
 listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
summary(ecarIIaw)
ecarIVaw <- spautolm(ft.SID74 ~ ft.NWBIR74, data=mdata.4,
 listw=sids.nhbr.listw.4, weights=BIR74, family="CAR")
summary(ecarIVaw)
nc.sids$fitIV <- append(fitted.values(ecarIVaw), NA, outl-1)
plot(nc.sids[,"fitIV"], nbreaks=12) # Cressie 1993, p. 565

## End(Not run)
## Not run:
data(oldcol, package="spdep")
COL.errW.eig <- errorsarlm(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::nb2listw(COL.nb, style="W"))
summary(COL.errW.eig)
COL.errW.sar <- spautolm(CRIME ~ INC + HOVAL, data=COL.OLD,
 spdep::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, spdep::nb2listw(boston.soi), family="SMA")
summary(gp1)

## End(Not run)
```

---

spBreg_lag *Bayesian MCMC spatial simultaneous autoregressive model estimation*

Description

The spBreg_lag function is an early-release version of the Matlab Spatial Econometrics Toolbox function sar_g.m, using drawing by inversion, and not accommodating heteroskedastic disturbances.

Usage

```
spBreg_lag(formula, data = list(), listw, na.action, Durbin, type,
    zero.policy=NULL, control=list())
spBreg_sac(formula, data = list(), listw, listw2=NULL, na.action,
    Durbin, type, zero.policy=NULL, control=list())
spBreg_err(formula, data = list(), listw, na.action, Durbin, etype,
    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)
```

Arguments

formula	a symbolic description of the model to be fit. The details of model specification are given for lm()
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
listw, listw2	a listw object created for example by nb2listw
na.action	a function (default options("na.action")), can also be na.omit or na.exclude with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to nb2listw may be subsetted.

`Durbin`	default FALSE (spatial lag model); if TRUE, full spatial Durbin model; if a formula object, the subset of explanatory variables to lag. From version 1.3-7, the presence of factors (categorical variables) in the Durbin term will give a warning, as it is as yet unknown how spatial lags of categorical variables should be interpreted.
`type`, `etype`	(use the 'Durbin=' argument - retained for backwards compatibility only) default "lag", may be set to "mixed"; when "mixed", the lagged intercept is dropped for spatial weights style "W", that is row-standardised weights, but otherwise included; "Durbin" may be used instead of "mixed"
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA
`control`	list of extra control arguments - see section below
`obj`	A spatial regression 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
`Q`	default NULL, else an integer number of cumulative power series impacts to calculate if `tr` is given

Details

From version 1.4.1, functions for models including spatially lagged independent variables warn on fitting if any of the right-hand side variables are factors. This is because the interpretation of coefficients that are not slopes is unclear when the variable is not interpretable on an unbounded line, such as factors. Factor variable names are shown with the suffix "(F)", others "dy/dx" in output from impact methods. A discussion can be found at `https://github.com/rsbivand/eqc25_talk`.

Control arguments

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

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

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

optimHessMethod: default "optimHess", may be "nlm" or one of the `optim` methods

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

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

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

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

MC_p: default 16; number of random variates

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

spamPivot: default “MMD”, alternative “RCM”

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

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

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

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

SE_method default “LU”, may be “MC”

nrho default 200, as in SE toolbox; the size of the first stage lndet grid; it may be reduced to for example 40

interpn default 2000, as in SE toolbox; the size of the second stage lndet grid

small_asy default TRUE; if the method is not “eigen”, use asymmetric covariances rather than numerical Hessian ones if n <= small

small default 1500; threshold number of observations for asymmetric covariances when the method is not “eigen”

SElndet default NULL, may be used to pass a pre-computed SE toolbox style matrix of coefficients and their lndet values to the "SE_classic" and "SE_whichMin" methods

LU_order default FALSE; used in “LU_prepermutate”, note warnings given for `lu` method

pre_eig default NULL; may be used to pass a pre-computed vector of eigenvalues

OrdVsign default 1; used to set the sign of the final component to negative if -1 (alpha times ((sigma squared) squared) in Ord (1975) equation B.1).

Extra Bayesian control arguments

ldet_method default “SE_classic”; equivalent to the `method` argument in `lagsarlm`

interval default `c(-1, 1)`; used unmodified or set internally by `jacobianSetup`

ndraw default `2500L`; integer total number of draws

nomit default `500L`; integer total number of omitted burn-in draws

thin default `1L`; integer thinning proportion

verbose default `FALSE`; inverse of `quiet` argument in `lagsarlm`

detval default `NULL`; not yet in use, precomputed matrix of log determinants

prior a list with the following components:

- **rhoMH, lambdaMH** default FALSE; use Metropolis or griddy Gibbs
- **Tbeta** default `NULL`; values of the betas variance-covariance matrix, set to `diag(k)*1e+12` if `NULL`
- **c_beta** default `NULL`; values of the betas set to 0 if `NULL`
- **rho** default `0.5`; value of the autoregressive coefficient
- **sige** default `1`; value of the residual variance
- **nu** default `0`; informative Gamma(nu,d0) prior on sige
- **d0** default `0`; informative Gamma(nu,d0) prior on sige
- **a1** default `1.01`; parameter for beta(a1,a2) prior on rho
- **a2** default `1.01`; parameter for beta(a1,a2) prior on rho
- **cc** default `0.2`; initial tuning parameter for M-H sampling
- **gG_sige** default TRUE; include sige in lambda griddy Gibbs update
- **cc1** default `0.2`; initial tuning parameter for M-H sampling
- **cc2** default `0.2`; initial tuning parameter for M-H sampling

Author(s)

Roger Bivand <`Roger.Bivand@nhh.no`>, with thanks to Abhirup Mallik and Virgilio Gómez-Rubio for initial coding GSoC 2011

References

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

Examples

```
#require("spdep", quietly=TRUE)
data(oldcol, package="spdep")
lw <- spdep::nb2listw(COL.nb, style="W")
require("coda", quietly=TRUE)
set.seed(1)
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))
## Not run:
ev <- eigenw(lw)
W <- as(lw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
 control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
 Durbin=TRUE)
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
 Durbin=TRUE, control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
 Durbin=~INC)
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.err.Bayes <- spBreg_err(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
 Durbin=~INC, control=list(prior=list(lambdaMH=TRUE)))
print(summary(COL.err.Bayes))
print(summary(impacts(COL.err.Bayes)))
print(raftery.diag(COL.err.Bayes, r=0.01))
set.seed(1)
COL.sacW.B0 <- spBreg_sac(CRIME ~ INC + HOVAL, data=COL.OLD, listw=lw,
 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=lw,
 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))
set.seed(1)
COL.lag.Bayes <- spBreg_lag(CRIME ~ INC + HOVAL, data=COL.OLD,
 listw=lw)
print(summary(COL.lag.Bayes))
print(summary(impacts(COL.lag.Bayes, tr=trMatc), short=TRUE, zstats=TRUE))
print(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=lw, Durbin=TRUE)
print(summary(COL.D0.Bayes))
print(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=lw, Durbin= ~ INC)
print(summary(COL.D1.Bayes))
print(summary(impacts(COL.D1.Bayes, tr=trMatc), short=TRUE, zstats=TRUE))
#data(elect80, package="spData")
#lw <- spdep::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")
#print(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)
```

```
#print(summary(el_B))
#print(el_ml$timings)
#print(attr(el_B, "timings"))

## End(Not run)
```

---

stsls *Generalized spatial two stage least squares*

Description

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

Usage

```
stsls(formula, data = list(), listw, zero.policy = NULL,
 na.action = na.fail, robust = FALSE, HC=NULL, legacy=FALSE, W2X = TRUE,
 sig2n_k=TRUE, adjust.n=FALSE)
## S3 method for class 'Stsls'
impacts(obj, ..., tr, R = NULL, listw = NULL, evalues=NULL, Q=NULL)
```

Arguments

`formula`	a symbolic description of the model to be fit. The details of model specification are given for `lm()`
`data`	an optional data frame containing the variables in the model. By default the variables are taken from the environment which the function is called.
`listw`	a `listw` object created for example by `nb2listw`
`zero.policy`	default NULL, use global option value; if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA - causing `lagsarlm()` to terminate with an error
`na.action`	a function (default `na.fail`), can also be `na.omit` or `na.exclude` with consequences for residuals and fitted values - in these cases the weights list will be subsetted to remove NAs in the data. It may be necessary to set zero.policy to TRUE because this subsetting may create no-neighbour observations. Note that only weights lists created without using the glist argument to `nb2listw` may be subsetted.
`robust`	default FALSE, if TRUE, apply a heteroskedasticity correction to the coefficients covariances
`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, overrides sig2n_k, 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; Python `spreg::GM_Lag` is default FALSE
`sig2n_k`	default TRUE - use n-k to calculate sigma^2, if FALSE use n; Python `spreg::GM_Lag` is default FALSE
`adjust.n`	default FALSE, used in creating spatial weights constants for the Anselin-Kelejian (1997) test
`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
`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
`Q`	default NULL, else an integer number of cumulative power series impacts to calculate if `tr` is given

Details

The fitting implementation fits a spatial lag model:

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

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

From version 1.3-6, the general Anselin-Kelejian (1997) test for residual spatial autocorrelation is added.

Value

an object of class "Stsls" containing:

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

Author(s)

Luc Anselin, Gianfranco Piras and Roger Bivand

References

Kelejian, H.H. and I.R. Prucha (1998). A generalized spatial two stage least squares procedure for estimating a spatial autoregressive model with autoregressive disturbances. *Journal of Real Estate Finance and Economics* 17, 99-121. doi:10.1023/A:1007707430416.

Anselin, L., & Kelejian, H. H. (1997). Testing for Spatial Error Autocorrelation in the Presence of Endogenous Regressors. International Regional Science Review, 20(1-2), 153-182. doi:10.1177/016001769702000109.

Roger Bivand, Gianfranco Piras (2015). Comparing Implementations of Estimation Methods for Spatial Econometrics. *Journal of Statistical Software*, 63(18), 1-36. doi:10.18637/jss.v063.i18.

See Also

lagsarlm

Examples

```
data(oldcol, package="spdep")
#require(spdep, quietly=TRUE)
lw <- spdep::nb2listw(COL.nb)
COL.lag.eig <- lagsarlm(CRIME ~ INC + HOVAL, data=COL.OLD, lw)
summary(COL.lag.eig, correlation=TRUE)
COL.lag.stsls <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, lw)
(x <- summary(COL.lag.stsls, correlation=TRUE))
coef(x)
W <- as(lw, "CsparseMatrix")
trMatc <- trW(W, type="mult")
loobj1 <- impacts(COL.lag.stsls, R=200, tr=trMatc)
summary(loobj1, zstats=TRUE, short=TRUE)
ev <- eigenw(lw)
loobj2 <- impacts(COL.lag.stsls, R=200, evalues=ev)
summary(loobj2, zstats=TRUE, short=TRUE)
require(coda)
HPDinterval(loobj1)
COL.lag.stslsW <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, lw, W2X=FALSE)
summary(COL.lag.stslsW, correlation=TRUE)
COL.lag.stslsWn <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, lw, W2X=FALSE, sig2n_k=FALSE)
summary(COL.lag.stslsWn, correlation=TRUE)
COL.lag.stslsR <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, lw,
robust=TRUE, W2X=FALSE)
summary(COL.lag.stslsR, correlation=TRUE)
COL.lag.stslsRl <- stsls(CRIME ~ INC + HOVAL, data=COL.OLD, lw,
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, spdep::nb2listw(boston.soi))
summary(gp2a)
```

trW *Spatial weights matrix powers traces*

Description

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

Usage

```
trW(W=NULL, m = 30, p = 16, type = "mult", listw=NULL, momentsSymmetry=TRUE)
mom_calc(lw, m)
mom_calc_int2(is, m, nb, weights, Card)
```

Arguments

`W`	A spatial weights matrix in CsparseMatrix form
`m`	The number of powers; must be an even number for 'type'="moments" (default changed from 100 to 30 (2010-11-17))
`p`	The number of samples used in Monte Carlo simulation of the traces if type is MC (default changed from 50 to 16 (2010-11-17))
`type`	Either "mult" (default) for powering a sparse matrix (with moderate or larger N, the matrix becomes dense, and may lead to swapping), or "MC" for Monte Carlo simulation of the traces (the first two simulated traces are replaced by their analytical equivalents), or "moments" to use the looping space saving algorithm proposed by Smirnov and Anselin (2009) - for "moments", `W` must be symmetric, for row-standardised weights through a similarity transformation
`listw`, `lw`	a listw object, which should either be fully symmetric, or be constructed as similar to symmetric from intrinsically symmetric neighbours using `similar.listw`, used with 'type'="moments"
`momentsSymmetry`	default TRUE; assert Smirnov/Anselin symmetry assumption
`is`	(used internally only in `mom_calc_int2` for 'type'="moments" on a cluster)
`nb`	(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)

Value

A numeric vector of `m` traces, with "timings" and "type" attributes; the 'type'="MC" also returns the standard deviation of the p-vector V divided by the square root of p as a measure of spread for the trace estimates.

Note

`mom_calc` and `mom_calc_int2` are for internal use only

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References

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

See Also

as_dgRMatrix_listw, nb2listw

Examples

```
require("sf", quietly=TRUE)
columbus <- st_read(system.file("shapes/columbus.gpkg", package="spData")[1], quiet=TRUE)
#require(spdep, quietly=TRUE)
col.gal.nb <- spdep::read.gal(system.file("weights/columbus.gal", package="spData")[1])
listw <- spdep::nb2listw(col.gal.nb)
W <- as(listw, "CsparseMatrix")
system.time(trMat <- trW(W, type="mult"))
str(trMat)
set.seed(1100)
system.time(trMC <- trW(W, type="MC"))
str(trMC)
plot(trMat, trMC)
abline(a=0, b=1)
for(i in 3:length(trMC)) {
 segments(trMat[i], trMC[i]-2*attr(trMC, "sd")[i], trMat[i],
  trMC[i]+2*attr(trMC, "sd")[i])
}
listwS <- similar.listw(listw)
W <- forceSymmetric(as(listwS, "CsparseMatrix"))
system.time(trmom <- trW(listw=listwS, m=24, type="moments"))
str(trmom)
all.equal(trMat[1:24], trmom, check.attributes=FALSE)
system.time(trMat <- trW(W, m=24, type="mult"))
str(trMat)
all.equal(trMat, trmom, check.attributes=FALSE)
set.seed(1)
system.time(trMC <- trW(W, m=24, type="MC"))
str(trMC)
## Not run:
data(boston, package="spData")
listw <- spdep::nb2listw(boston.soi)
listwS <- similar.listw(listw)
system.time(trmom <- trW(listw=listwS, m=24, type="moments"))
str(trmom)
library(parallel)
nc <- max(2L, detectCores(logical=FALSE), na.rm = TRUE)-1L
```

```
# set nc to 1L here
if (nc > 1L) nc <- 1L
coresOpt <- get.coresOption()
invisible(set.coresOption(nc))
if(!get.mcOption()) {
  cl <- makeCluster(get.coresOption())
  set.ClusterOption(cl)
}
system.time(trmomp <- trW(listw=listwS, m=24, type="moments"))
if(!get.mcOption()) {
  set.ClusterOption(NULL)
  stopCluster(cl)
}
all.equal(trmom, trmomp, check.attributes=FALSE)
invisible(set.coresOption(coresOpt))

## End(Not run)
```

Index