Package ‘mgwrsar’

April 17, 2023

Type Package
Title GWR and MGWR with Spatial Autocorrelation
Version 1.0.4
Date 2023-04-14
Author Ghislain Geniaux and Davide Martinetti
Maintainer Ghislain Geniaux <ghislain.geniaux@inrae.fr>
License GPL (>= 2)
Depends R (>= 3.5.0), Rcpp, sp, leaflet, Matrix
Imports ggplot2, sf, knitr, spgwr, methods, doParallel, foreach, htmltools, nabor, qlcMatrix, mapview, microbenchmark, rlang, dplyr, gridExtra, grid, mboost, mgcv, caret
VignetteBuilder knitr
LinkingTo RcppEigen, Rcpp
RoxygenNote 7.2.3
NeedsCompilation yes
Encoding UTF-8
Repository CRAN
Date/Publication 2023-04-17 08:50:01 UTC

R topics documented:

bandwidths_mgwrsar ........................................ 2
find_TP ......................................................... 4
kernel_matW .................................................. 5
MGWRSAR .......................................................... 7
mgwrsar_bootstrap_test ..................................... 10
mgwrsar_bootstrap_test_all ................................ 11
multiscale_gwr .............................................. 11
bandwidths_mgwrsar

Select optimal kernel and bandwidth from a list of models, kernels and bandwidth candidates.

Description

Given a lm formula and a dataframe with coordinates, function bandwidths_mgwrsar optimizes the choice of a bandwidth value for each of the chosen models and kernel types using a leave-one-out cross validation criteria. A cross validated criteria is also used for selecting the best kernel type for a given model.

Usage

bandwidths_mgwrsar(formula, data, coord, 
fixed_vars='Intercept', Models='GWR', candidates_Kernels='bisq', 
control=list(), control_search=list())

Arguments

formula a formula.

data a dataframe or a spatial dataframe (sp package).

coord a dataframe or a matrix with coordinates, not required if data is a spatial dataframe, default NULL.

fixed_vars a vector with the names of spatially constant coefficient. For mixed model, if NULL, the default # is set to 'Intercept'.

Models character containing the type of model: Possible values are "OLS", "SAR", "GWR" (default), "MGWR", "MGWRSAR_0_0_kv","MGWRSAR_1_0_kv", "MGWRSAR_0_kc_kv", "MGWRSAR_1_kc_kv", "MGWRSAR_1_kc_0".

candidates_Kernels a vector with the names of kernel type.

control list of extra control arguments for MGWRSAR wrapper - see MGWRSAR help.

control_search list of extra control arguments for bandwidth/kernel search - see section below.

@details

• search_Wif TRUE select an optimal spatial weight matrix using a moment estimator, default FALSE.
 bandwidths_mgwrsar

- kernels_wif search_W is TRUE, kernels_w is a vector of candidated kernels types, default NULL.
- lower_clower bound for bandwidth search (default, the approximate first decile of distances).
- upper_cupper bound for bandwidth search (default, the approximate last decile of distances).
- lower_dlower bound for discrete kernels, default 2*k+1.
- lower_dWower bound for discrete kernels for finding optimal spatial weight matrix, default 2.
- lower_cWlower bound for bandwidth search for finding optimal spatial weight matrix (default approximate 0.005 quantile of distances).

Details

Given a lm formula and a dataframe with coordinates, for each model in Models for which a bandwidth is required, this function optimizes the choice of a bandwidth value for each of the chosen models and kernel types using a leave one out cross validation criteria. A cross validated criteria is also used for selecting the best kernel type for a given model.

Value

bandwidths_MGWRSA  returns a list with:

config_model a vector with information about model, optimal kernel and bandwidth for local regression, and optimal kernel and bandwidth for spatial weight matrix W.

SSR The sum of square residuals.

CV The CV criteria.

model objects of class mgwrsar estimated using config_model

References


See Also

MGWRSA, summary_mgwrsar, plot_mgwrsar, predict_mgwrsar
find_TP

Search of a suitable set of target points. find_TP is a wrapper function that identifies a set of target points based on spatial smoothed OLS residuals.

Usage

find_TP(formula, data, coord, K, kWtp=16, Wtp=NULL, type='residuals', model_residuals=NULL, verbose=0, prev_TP=NULL, nTP=NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>a formula</td>
</tr>
<tr>
<td>data</td>
<td>a dataframe or a spatial dataframe (SP package)</td>
</tr>
<tr>
<td>coord</td>
<td>a dataframe or a matrix with coordinates, not required if data is a spatial dataframe</td>
</tr>
<tr>
<td>K</td>
<td>the minimum number of first neighbors with lower (resp. higher) absolute value of the smoothed residuals.</td>
</tr>
<tr>
<td>kWtp</td>
<td>the number of first neighbors for computing the smoothed residuals, default 16.</td>
</tr>
<tr>
<td>Wtp</td>
<td>a precomputed matrix of weights, default NULL.</td>
</tr>
<tr>
<td>type</td>
<td>method for choosing TP, could be 'residuals', 'equidistantGrid', 'random', default 'residuals'</td>
</tr>
</tbody>
</table>
model_residuals
(optional) a vector of residuals.

verbose
verbose mode, default FALSE.

prev_TP
index of already used TP (version length(K)>1), default NULL.

nTP
number of target points for random choice of target points, default NULL.

Details

find_TP is a wrapper function that identifies a set of target points, based on spatial smoothed residuals by default. If no vector of residuals are provided, OLS residuals are computed. The function first computes the smooth of model residuals using a Sheppard’s kernel with kWtp neighbors (default 16). Then it identifies local maxima (resp. minima) that fits the requirement of having at least K neighbors with lower (resp. higher) absolute value of the smoothed residuals. As K increases the number of target points decreases.

Value

find_TP returns an index vector of Target Points set.

Examples

```r
library(mgwrsar)
## loading data example
data(mydata)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
TP=find_TP(formula = \'Y_gwr~X1+X2+X3\', data =mydata,coord=coord,K=6,type='residuals')
# only 60 targets points are used
length(TP)

model_GWR_tp<-MGWRSAR(formula = \'Y_gwr~X1+X2+X3\', data = mydata,coord=coord,
fixed_vars=NULL,kernels=c('gauss'), H=0.03, Model = 'GWR',
control=list(SE=TRUE,TP=TP,kWtp=12))
summary(model_GWR_tp$Betav)
```

kernel_matW

A function that returns a sparse weight matrix based computed with a specified kernel (gauss,bisq,tcub,epane,rectangle,triangle) considering coordinates provides in S and a given bandwidth. If NN<row(S) only NN firsts neighbours are considered. If Type!='GD' then S should have additional columns and several kernels and bandwidths should be specified by the user.
**Description**

`kernel_matW` is a function that returns a sparse weight matrix based on a specified kernel (gauss, bisq, tsub, epane, rectangle, triangle) considering the coordinates provided in S and a given bandwidth. If `NN < nrow(S)` only `NN` first neighbors are considered. If `Type != 'GD'` then `S` should have additional columns and several kernels and bandwidths should be specified by the user.

**Usage**

```r
kernel_matW(H, kernels, coord_i, coord_j = NULL, NN, ncolX = 1, 
Type = 'GD', adaptive = F, diagnull = TRUE, rowNorm = TRUE, noisland = FALSE)
```

**Arguments**

- **H**: A vector of bandwidths
- **kernels**: A vector of kernel types
- **coord_i**: A matrix with variables used in kernel (reference)
- **coord_j**: A matrix with variables used in kernel (neighbors), default NULL (if NULL `coord_j = coord_i`)
- **NN**: Number of spatial Neighbours for kernels computations
- **ncolX**: control parameter
- **Type**: Type of Generalized kernel product ("GD" only spatial, "GDC" spatial + a categorical variable, "GDX" spatial + a continuous variable, "GDT" spatial + a time index, and other combinations "GDXXC", "GDTX", ...)
- **adaptive**: A vector of boolean to choose adaptive version for each kernel
- **diagnull**: Zero on diagonal, default FALSE
- **rowNorm**: A boolean, row normalization of weights, default TRUE
- **noisland**: A boolean to avoid isle with no neighbors for non adaptive kernel, default FALSE

**Value**

A sparse Matrix of weights (dgCMatrix).

**Examples**

```r
library(mgwrsar)
## loading data example
data(mydata)
coord = as.matrix(mydata[,c("x_lat","y_lon")])
## Creating a spatial weight matrix (sparse dgCMatrix) of 4 nearest neighbors with 0 in diagonal
W = kernel_matW(H = 4, kernels = 'rectangle', coord_i = coord, NN = 4, adaptive = TRUE, diagnull = TRUE, rowNorm = TRUE)
```
MGWRSAR

Estimation of linear and local linear model with spatial autocorrelation model (mgwrsar).

Description

MGWRSAR is a wrapper function for estimating linear and local linear models with spatial autocorrelation (SAR models with spatially varying coefficients).

Usage

MGWRSAR(formula, data, coord, fixed_vars=NULL, kernels, H, Model='GWR', control=list())

Arguments

- formula: a formula.
- data: a dataframe or a spatial dataframe (sp package).
- coord: default NULL, a dataframe or a matrix with coordinates, not required if data is a spatial dataframe.
- fixed_vars: a vector with the names of spatially constant coefficient for mixed model. All other variables present in formula are supposed to be spatially varying. If empty or NULL (default), all variables in formula are supposed to be spatially varying.
- kernels: A vector containing the kernel types. Possible types: rectangle ("rectangle"), bisquare ("bisq"), tricube ("tcub"), epanechnikov ("epane"), gaussian ("gauss").
- H: vector containing the bandwidth parameters for the kernel functions.
- Model: character containing the type of model: Possible values are "OLS", "SAR", "GWR" (default), "MGWR", "MGWRSAR_0_0_kv", "MGWRSAR_1_0_kv", "MGWRSAR_0_kc_kv", "MGWRSAR_1_kc_kv", "MGWRSAR_1_kc_0". See Details for more explanation.
- control: list of extra control arguments for MGWRSAR wrapper - see Details below

Details

- Z: a matrix of variables for generalized kernel product, default NULL.
- W: a row-standardized spatial weight matrix for Spatial Autocorrelation, default NULL.
- type: verbose mode, default FALSE.
- adaptive: a vector of boolean to choose adaptive version for each kernel.
- kernel_w: the type of kernel for computing W, default NULL.
- h_w: the bandwidth value for computing W, default 0.
- Method: estimation technique for computing the models with Spatial Dependence. '2SLS' or 'B2SLS', default '2SLS'.

Description

MGWRSAR is a wrapper function for estimating linear and local linear models with spatial autocorrelation (SAR models with spatially varying coefficients).

Usage

MGWRSAR(formula, data, coord, fixed_vars=NULL, kernels, H, Model='GWR', control=list())

Arguments

- formula: a formula.
- data: a dataframe or a spatial dataframe (sp package).
- coord: default NULL, a dataframe or a matrix with coordinates, not required if data is a spatial dataframe.
- fixed_vars: a vector with the names of spatially constant coefficient for mixed model. All other variables present in formula are supposed to be spatially varying. If empty or NULL (default), all variables in formula are supposed to be spatially varying.
- kernels: A vector containing the kernel types. Possible types: rectangle ("rectangle"), bisquare ("bisq"), tricube ("tcub"), epanechnikov ("epane"), gaussian ("gauss").
- H: vector containing the bandwidth parameters for the kernel functions.
- Model: character containing the type of model: Possible values are "OLS", "SAR", "GWR" (default), "MGWR", "MGWRSAR_0_0_kv", "MGWRSAR_1_0_kv", "MGWRSAR_0_kc_kv", "MGWRSAR_1_kc_kv", "MGWRSAR_1_kc_0". See Details for more explanation.
- control: list of extra control arguments for MGWRSAR wrapper - see Details below

Details

- Z: a matrix of variables for generalized kernel product, default NULL.
- W: a row-standardized spatial weight matrix for Spatial Autocorrelation, default NULL.
- type: verbose mode, default FALSE.
- adaptive: a vector of boolean to choose adaptive version for each kernel.
- kernel_w: the type of kernel for computing W, default NULL.
- h_w: the bandwidth value for computing W, default 0.
- Method: estimation technique for computing the models with Spatial Dependence. '2SLS' or 'B2SLS', default '2SLS'.
• TP A vector of target points, default NULL.
• isgcv computing LOOCV criteria (for example for selecting optimal bandwidth), default FALSE.
• isfgcv if TRUE, simplify the computation of CV criteria (remove or not i when using local instruments for model with lambda spatially varying), default TRUE.
• maxknn when n > NmaxDist, only the maxknn first neighbours are used for distance computation, default 500.
• NmaxDist when n > NmaxDist only the maxknn first neighbours are used for distance computation, default 5000
• verbose verbose mode, default FALSE.

Value

MGWRSAR returns an object of class mgwrsar with at least the following components:

- Betav: matrix of coefficients of dim(n, kv) x kv.
- Betac: vector of coefficients of length kc.
- Model: The sum of square residuals.
- Y: The dependent variable.
- XC: The explanatory variables with constant coefficients.
- XV: The explanatory variables with varying coefficients.
- X: The explanatory variables.
- W: The spatial weight matrix for spatial dependence.
- isgcv: if gcv has been computed.
- edf: The estimated degrees of freedom.
- formula: The formula.
- data: The dataframe used for computation.
- Method: The type of model.
- coord: The spatial coordinates of observations.
- H: The bandwidth vector.
- fixed_vars: The names of constant coefficients.
- kernels: The kernel vector.
- SSR: The sum of square residuals.
- residuals: The vector of residuals.
- fit: the vector of fitted values.
- sev: local standard error of parameters.
- NN: Maximum number of neighbors for weights computation.
MGWRSAR is a wrapper function for estimating linear and local linear model with spatial autocorrelation that allows to estimate the following models:

\[ y = \beta_c X_c + \epsilon \] (OLS)

\[ y = \beta_c X_c + \beta_v(u_i, v_i)X_v + \epsilon \] (GWR)

\[ y = \lambdaWy + \beta_c X_c + \epsilon \] (MGWR-SAR(0,k,0))

\[ y = \lambdaWy + \beta_c(u_i, v_i)X_v + \epsilon \] (MGWR-SAR(0,0,k))

\[ y = \lambdaWy + \beta_c X_c + \beta_v(u_i, v_i)X_v + \epsilon \] (MGWR-SAR(0,k_c,k_v))

\[ y = \lambda(u_i, v_i)Wy + \beta_c X_c + \epsilon \] (MGWR-SAR(1,k,0))

\[ y = \lambda(u_i, v_i)Wy + \beta_v(u_i, v_i)X_v + \epsilon \] (MGWR-SAR(1,0,k))

\[ y = \lambda(u_i, v_i)Wy + \beta_c X_c + \beta_v(u_i, v_i)X_v + \epsilon \] (MGWR-SAR(1,k_c,k_v))

When model imply spatial autocorrelation, a row normalized spatial weight matrix must be provided. 2SLS and Best 2SLS method can be used. When model imply local regression, a bandwidth and a kernel type must be provided. Optimal bandwidth can be estimated using bandwidths_mgwrsar function. When model imply mixed local regression, the names of stationary covariates must be provided.

In addition to the ability of considering spatial autocorrelation in GWR/MGWR like models, MGWRSAR function introduces several useful technics for estimating local regression with space coordinates:

- it uses RCCP and RCCPeigen code that speed up computation and allows parallel computing via doMC package;
- it allows to drop out variables with not enough local variance in local regression, which allows to consider dummies in GWR/MGWR framework without trouble.
- it allows to drop out local outliers in local regression.
- it allows to consider additional variable for kernel, including time (asymmetric kernel) and categorical variables (see Li and Racine 2010). Experimental version.

References


See Also

bandwidths_mgwrsar, summary_mgwrsar, plot_mgwrsar, predict_mgwrsar, kernel_matW
Examples

```r
library(mgwrsar)
## loading data example
data(mydata)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
## Creating a spatial weight matrix (sparse dgCMatrix)
## of 4 nearest neighbors with 0 in diagonal
W=kernel_matW(H=4,kernels='rectangle',coord_i=coord,NN=4,adaptive=TRUE,
diagnull=TRUE,rowNorm=TRUE)
mgwrsar_0_kc_kv<-MGWRSAR(formula = Y_mgwrsar_0_kc_kv~X1+X2+X3,
data = mydata,
coord=coord, fixed_vars='X2',kernels=c('gauss'),H=20, Model = 'MGWRSAR_0_kc_kv',
control=list(SE=FALSE,adaptive=TRUE,W=W))
summary_mgwrsar(mgwrsar_0_kc_kv)
```

mgwrsar_bootstrap_test

* A bootstrap test for Betas for mgwrsar class model.

Description

A bootstrap test for Betas for mgwrsar class model.

Usage

```r
mgwrsar_bootstrap_test(x0,x1,B=100,domc=FALSE,ncore=1,
type='standard',eps='H1',df='H1',focal='median',D=NULL)
```

Arguments

- `x0`: The H0 mgwrsar model
- `x1`: The H1 mgwrsar model
- `B`: number of bootstrap repetitions, default 100
- `domc`: If TRUE, doParallel parallelization
- `ncore`: number of cores
- `type`: type of bootstrap : 'wild','Rademacher','spatial' or 'standard' (default)
- `eps`: Hypothesis under wich residuals are simulated, 'H0' or 'H1' (default)
- `df`: Hypothesis under wich degree of freedom is estimated.
- `focal`: see sample_stat help
- `D`: A matrix of distance

Value

The value of the statistics test and a p ratio.
**Description**

A bootstrap test for testing nullity of all Betas for mgwrsar class model,

**Usage**

```r
mgwrsar_bootstrap_test_all(model,B=100,domc=NULL)
```

**Arguments**

- `model`: A mgwrsar model
- `B`: number of bootstrap replications, default 100
- `domc`: If TRUE, doMC parallelization

**Value**

a matrix with statistical test values and p ratios

**See Also**

- `mgwrsar_bootstrap_test`
Arguments

- formula: a formula.
- data: a dataframe or a spatial dataframe (sp package).
- coord: default NULL, a dataframe or a matrix with coordinates, not required if data is a spatial dataframe.
- fixed_vars: a vector with the names of spatial lay constant coefficient for mixed model. All other variables present in formula are supposed to be spatially varying. If empty or NULL (default), all variables in formula are supposed to be spatially varying.
- Model: character containing the type of model: Possible values are "OLS", "SAR", "GWR" (default), "MGWR", "MGWRSAR_0_0_kv","MGWRSAR_1_0_kv", "MGWRSAR_0_kc_kv","MGWRSAR_1_kc_kv", "MGWRSAR_1_kc_0". See Details for more explanation.
- kernels: A vector containing the kernel types. Possible types: rectangle ("rectangle"), bisquare ("bisq"), tricube ("tcub"), epanechnikov ("epane"), gaussian ("gauss")
- maxiter: maximum number of backfitting iteration
- control: list of extra control arguments for MGWRSAR wrapper - see Details below
- fd: to be documented (experimental)
- H: vector containing the bandwidth parameters for each covariate.
- BETA: to be documented (experimental)
- grad: to be documented (experimental)
- nstable: to be documented (experimental)
- nfull: to be documented (experimental)
- init: to be documented (experimental)

Description

multiscale_gwr.cv to be documented (experimental)

Usage

multiscale_gwr.cv(dataName, argDataName="data", target='Y', K=5, regFun, par_model, par_model2=NULL, regFun2=NULL, predFun, args_predNames, extra_args_pred=NULL, namesXtraArgs2Split=NULL, myseed=1)
mydata

Arguments

dataName character, name of the data
argDataName character, generic name to use as data name.
target character, name of variable to explain
K integer, number of folds for cross validation
regFun character, name of the estimation function
par_model named list with the arguments for the estimation function
par_model2 to be documented
regFun2 to be documented
predFun character, name of the prediction function
args_predNames named list with the arguments for the prediction function
extra_args_pred named list with extra arguments for non generic prediction function
namesXtraArgs2Split character, names of the objects in extra_args_pred that need to be split for cross validation.
myseed seed for random number.

mydata is a simulated data set of a mgwrsar model

Description

mydata is a simulated data set of a mgwrsar model

Author(s)

Ghislain Geniaux and Davide Martinetti <ghislain.geniaux@inra.fr>

References

**normW**

*normW row normalization of dgCMatrix*

**Description**

normW row normalization of dgCMatrix

**Usage**

`normW(W)`

**Arguments**

`W` A dgCMatrix class matrix

**Value**

A row normalized dgCMatrix

---

**plot_effect**

*plot_effect plot_effect is a function that plots the effect of a variable X_k with spatially varying coefficient, i.e X_k * Beta_k(u_i,v_i) for comparing the magnitude of effects of between variables.*

**Description**

plot_effect plot_effect is a function that plots the effect of a variable X_k with spatially varying coefficient, i.e X_k * Beta_k(u_i,v_i) for comparing the magnitude of effects of between variables.

**Usage**

`plot_effect(model,sampling=TRUE,nsample=2000,nsample_max=5000,title='')`

**Arguments**

`model` a model of mgwrsar class with some spatially varying coefficients.

`sampling` Boolean, if nrow(model$Betav)> nsample_max a sample of size nsample is randomly selected, default TRUE.

`nsample` integer, size of the sample if sampling is TRUE, default 2000.

`nsample_max` integer, size max to engage sampling if sampling is TRUE, default 5000.

`title` a title for the plot.
plot_mgwrsa

Examples

library(mgwrsa)
## loading data example
data(mydata)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
## Creating a spatial weight matrix (sparse dgCMMatrix)
## of 8 nearest neighbors with 0 in diagonal
model_GWR0<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata,coord=coord,
fixed_vars=NULL,kernels=c('gauss'),H=0.13, Model = 'GWR',control=list(SE=TRUE))
plot_effect(model_GWR0)

plot_mgwrsa

plot_mgwrsa plots the values of local parameters of a mgwrsa models using a leaflet map.

Description

plot_mgwrsa plots the values of local parameters of a mgwrsa models using a leaflet map.

Usage

plot_mgwrsa(model,type='coef',var=NULL,crs=NULL,mypalette = "RdYlGn",opacity=0.5
,fopacity=0.5,nbins=8,radius=500,mytile='Stamen.TonerBackground',myzoom=8,
myresolution=150,LayersControl=TRUE,myzoomControl=TRUE,mytile2=NULL,ScaleBar=NULL,
ScaleBarOptions=list(maxWidth = 200, metric = TRUE,imperial = FALSE,
updateWhenIdle = TRUE),MyLegendTitle=NULL,lopacity=0.5)

Arguments

model a mgwrsa model.
type default 'coef', for plotting the value of the coefficients. Local t-Student could also be plot using 't_coef', residuals using 'residuals' and fitted using 'fitted'.
var Names of variable to plot.
crs A CRS projection.
mypalette A leaflet palette.
opacity Opacity of border color.
fopacity Opacity of fill color.
nbins nbins.
radius radius of circle for plot of points.
mytile tile 1.
myzoom level of zoom for tile 1.
myresolution resolution for tile 1.
predict_mgwrsar

LayersControl  layers controls.
myzoomControl  zoom control.
mytile2  tile 2.
ScaleBar  ScaleBar.
ScaleBarOptions  options for ScaleBar.
MyLegendTitle  Legend title.
opacity  opacity for legend.

Value

A Interactive Web Maps with local parameters plot and Open Street Map layer.

See Also

MGWRSAR, bandwidths_mgwrsar, summary_mgwrsar, predict_mgwrsar, kernel_matW

Examples

library(mgwrsar)
## loading data example
data(mydata)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
## Creating a spatial weight matrix (sparse dgCMatrix)
## of 4 nearest neighbors with 0 in diagonal
model_GWR0<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata,coord=coord,
fixed_vars=NULL,kernels=c('gauss'),H=0.13, Model='GWR',control=list(SE=TRUE))
summary_mgwrsar(model_GWR0)
plot_mgwrsar(model_GWR0,type="B_coef",var='X2')
plot_mgwrsar(model_GWR0,type="t_coef",var='X2')

predict_mgwrsar

mgwrsar Model Predictions predict_mgwrsar is a function for computing predictions of a mgwrsar models. It uses Best Linear Unbiased
Predictor for mgwrsar models with spatial autocorrelation.

Description

mgwrsar Model Predictions predict_mgwrsar is a function for computing predictions of a mgwrsar models. It uses Best Linear Unbiased Predictor for mgwrsar models with spatial autocorrelation.

Usage

predict_mgwrsar(model, newdata, newdata_coord, W = NULL, type = "BPN",
h_w = 100, kernel_w = "rectangle", maxobs=4000, beta_proj=FALSE,
method_pred='TP', k_extra = 8)
The `predict_mgwrsar` function is used to make predictions from a model of the `mgwrsar` class. Here are the key arguments and details:

**Arguments**

- **model**: a model of `mgwrsar` class.
- **newdata**: a matrix or data.frame of new data.
- **newdata_coord**: a matrix of new coordinates, and eventually other variables if a General Kernel Product is used.
- **W**: the spatial weight matrix for models with spatial autocorrelation.
- **type**: Type for BLUP estimator, default "BPN". If NULL use predictions without spatial bias correction.
- **h_w**: A bandwidth value for the spatial weight matrix.
- **kernel_w**: kernel type for the spatial weight matrix. Possible types: rectangle ("rectangle"), bisquare ("bisq"), tricube ("tcub"), epanechnikov ("epane"), or gaussian ("gauss").
- **maxobs**: maximum number of observations for exact calculation of solve(I- rho*W), default maxobs=4000.
- **beta_proj**: A boolean, if TRUE the function then return a two elements list(Y_predicted,Beta_proj_out).
- **method_pred**: If method_pred = "TP" (default) prediction is done by recomputing a MGWRSAR model with new-data as target points, else if method_pred in ("tWtp_model","model","sheppard") a matrix for projecting estimated betas is used (see details).
- **k_extra**: number of neighboors for local parameter extrapolation if sheppard kernel is used, default 8.

**Details**

If method_pred='tWtp_model', the weighting matrix for prediction is based on the expected weights of outsample data if they were had been added to insample data to estimate the corresponding MGWRSAR (see Geniaux 2022 for further detail), if method_pred = sheppard a sheppard kernel with k_extra neighbours (default 8) is used and if method_pred='kernel_model' the same kernel and number of neighbors as for computing the MGWRSAR model is used.

**Value**

A vector of predictions if beta_proj is FALSE or a list with a vector named Y_predicted and a matrix named Beta_proj_out.

**See Also**

`MGWRSAR`, `bandwidths_mgwrsar`, `summary_mgwrsar`, `plot_mgwrsar`, `kernel_matW`

**Examples**

```r
library(mgwrsar)
data(mydata)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
length_out=800
index_in=sample(1:1000,length_out)
```
summary_Matrix

```r
index_out=(1:1000)[-index_in]
model_GWR_insample<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata[index_in,],
coord=coord[index_in,],fixed_vars=NULL,kernels=c('gauss'),H=8, Model = 'GWR',
control=list(adaptive=TRUE))
summary_mgwrsar(model_GWR_insample)

newdata=mydata[index_out,]
newdata_coord=coord[index_out,]
newdata$Y_mgwrsar_1_0_kv=0
Y_pred=predict_mgwrsar(model_GWR_insample, newdata=newdata,
newdata_coord=newdata_coord)
head(Y_pred)
head(mydata$Y_gwr[index_out])
sqrt(mean((mydata$Y_gwr[index_out]-Y_pred)^2)) # RMSE
```

---

**summary_Matrix**

**summary_Matrix to be documented**

### Description

summary_Matrix to be documented

### Usage

```
summary_Matrix(object, ...)
```

### Arguments

- **object**
  - to be documented
- **...**
  - to be documented

### Value

- to be documented
summary_mgwrsar

Print a summary of mgwrsar models

Description
Print a summary of mgwrsar models

Usage
summary_mgwrsar(model)

Arguments
model a model of class mgwrsar

Value
a summary of mgwrsar models

See Also
MGWRSAR, bandwidths_mgwrsar, plot_mgwrsar, predict_mgwrsar, kernel_matW

Examples

library(mgwrsar)
## loading data example
data(mydata)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
## Creating a spatial weight matrix (sparse dgCMatrix)
## of 4 nearest neighbors with 0 in diagonal
W=kernel_matW(H=4,kernels='rectangle',coord_i=coord,NN=4,adaptive=TRUE,
diagnull=TRUE,rowNorm=TRUE)
mgwrsar_0_kc_kv<-MGWRSAR(formula = 'Y_mgwrsar_0_kc_kv~X1+X2+X3', data = mydata,
coord=coord, fixed_vars='X2',kernels=c('gauss'),H=20, Model = 'MGWRSAR_0_kc_kv',
control=list(SE=FALSE,adaptive=TRUE,W=W))
summary_mgwrsar(mgwrsar_0_kc_kv)
Index

bandwidths_mgwrsar, 2
find_TP, 4
kernel_matW, 5
MGWRSAR, 7
mgwrsar_bootstrap_test, 10
mgwrsar_bootstrap_test_all, 11
multiscale_gwr, 11
multiscale_gwr.cv, 12
mydata, 13

normW, 14
plot_effect, 14
plot_mgwrsar, 15
predict_mgwrsar, 16

summary_Matrix, 18
summary_mgwrsar, 19