Package ‘mgwrsar’

May 11, 2018

Type Package
Title GWR and MGWR with Spatial Autocorrelation
Version 0.1
Date 2018-05-05
Author Ghislain Geniaux and Davide Martinetti
Maintainer Ghislain Geniaux <ghislain.geniaux@inra.fr>
License GPL (>= 2)
Depends Rcpp, sp, leaflet, Matrix
Imports spgwr, methods, doParallel, foreach, htmltools, nabor
Suggests knitr, rmarkdown
VignetteBuilder knitr
LinkingTo RcppEigen, Rcpp
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2018-05-11 10:44:14 UTC

R topics documented:

mgwrsar-package ......................................................... 2
bandwidths_mgwrsar .................................................. 4
bisq ................................................................. 7
bisq_C ............................................................... 8
bisq_knn_C ............................................................ 9
gauss_adapt .......................................................... 9
gauss_adapt_C ....................................................... 10
kernelW_C ............................................................ 11
KNN ................................................................. 12
mgwrsar-package

MGWRSAR ........................................................................ 12
mgwrsar_bootstrap_test ...................................................... 16
mgwrsar_bootstrap_test_all ............................................... 17
mydata ............................................................................ 17
plot_mgwrsar .................................................................. 18
predict_mgwrsar ............................................................... 19
summary_mgwrsar ............................................................. 20

Index 22

mgwrsar-package GWR and MGWR with Spatial Autocorrelation

Description

mgwrsar package proposes functions for estimating linear and local linear model with spatial autocorrelation. It allows to estimate linear and Spatial Autoregressive models with spatially varying coefficients. Models that mixed spatially varying and stationary coefficients can also be estimated.

Package: mgwrsar
Type: Package
Title: GWR and MGWR with Spatial Autocorrelation
Version: 0.1
Date: 2018-05-05
Author: Ghislain Geniaux and Davide Martinetti
Maintainer: Ghislain Geniaux <ghislain.geniaux@inra.fr>
License: GPL (>= 2)
Depends: Rcpp, sp, leaflet, Matrix
Imports: spgwr, methods, doParallel, foreach, htmltools, nabor
Suggests: knitr, rmarkdown
VignetteBuilder: knitr
LinkingTo: RcppEigen, Rcpp
RoxygenNote: 6.0.1

Index of help topics:

KNN A function that returns a row normalized weight matrix based on k first neighbors, to be documented
MGWRSAR Estimation of linear and local linear model with spatial autocorrelation model (mgwrsar).
bandwidths_mgwrsar Select optimal kernel and bandwidth from a list of models, kernels and bandwidth candidates.
bisq bisquare kernel
bisq_C bisquare kernel, RcppEigen version
bisq_knn_C Adaptive bisquare kernel
gauss_adapt Adaptive gaussian kernel
gauss_adapt_C Adaptive gaussian kernel, RcppEigen version
kernelw_C Computes weight matrix for a given kernel and bandwidth
mgwrsar-package GWR and MGWR with Spatial Autocorrelation
mgwrsar_bootstrap_test A bootstrap test for Betas for mgwrsar class model.
mgwrsar_bootstrap_test_all A bootstrap test for testing nullity of all Betas for mgwrsar class model,
mydata mydata is a simulated data set of a mgwrsar model
plot_mgwrsar plot_mgwrsar plots the value of local parameters of a mgwrsar models using a leaflet map.
predict_mgwrsar mgwrsar Model Predictions
summary_mgwrsar Print a summary of mgwrsar models

Further information is available in the following vignettes:

mgwrsar-basic_examples Examples of basic uses of mgwrsar package (source, pdf)

Details

Author(s)
Ghislain Geniaux <ghislain.geniaux hat inra.fr> Davide Martinetti <davide.martinetti hat inra.fr>

References

See Also
lwr locfit

Examples
library(mgwrsar)
## loading data example
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', 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".
- **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

```r
data(data_mgwrsar)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
## Creating a spatial weight matrix (sparse dgCMatrix) of 8 nearest neighbors
W=KNN(coord,8)

timu<-proc.time()
model_GWR<-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,doMC=FALSE))
(proc.time()-timu)[3]

summary_mgwrsar(model_GWR)
plot_mgwrsar(model_GWR,type='B_coef',var='X2')
plot_mgwrsar(model_GWR,type='t_coef',var='X2')
```
bandwidths_mgwrsar

- search_Wif TRUE select an optimal spatial weight matrix using a moment estimator, default FALSE.
- kernels_wif search_W is TRUE, kernels_w is a vector of candidate kernel 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_MGWRSAR 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

MGWRSAR, summary_mgwrsar, plot_mgwrsar, predict_mgwrsar, kernelW_C
Examples

```r
library(mgwr)
data(data_mgwr)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
W=KNN(coord,8)

## Finding bandwidth by hand

## kernel only space
model_GWR<-MGWR(formula = 'Y_gwr~X1+X2+X3', data = mydata, coord=coord, fixed_vars=NULL,kernels=c('bisq_knn'),H=50, Model = 'GWR', control=list(isgcv=FALSE,minv=1))
cat('SSR =')
summary_mgwr(model_GWR)

myCV<-function(H)(model_GWR<-MGWR(formula = 'Y_gwr~X1+X2+X3', data = mydata, coord=coord, fixed_vars=NULL,kernels=c('bisq_knn'),H=H, Model = 'GWR',control=list(isgcv=TRUE))
model_GWR$SSR
}
res=optimize(myCV,upper=500,lower=10)
res

## model with optimal bandwith with adaptative gaussian kernel

model_GWR<-MGWR(formula = 'Y_gwr~X1+X2+X3', data = mydata, coord=coord, fixed_vars=NULL,kernels=c('gauss_adapt'),H=ceiling(res$minimum), Model = 'GWR',control=list(isgcv=FALSE))
summary_mgwr(model_GWR)

## finding the bandwiths using bandwidths_mgwr

mytab<-bandwidths_mgwr(formula = 'Y_gwr~X1+X2+X3', data = mydata,coord=coord, fixed_vars='Intercept',Models=c('GWR','MGWR'),Kernels=c('bisq_knn','gauss_adapt','gauss'), control=list(),control_search=list (lower_d=8,lower_c=0.03,upper_c=0.65))

names(mytab)
names(mytab[['GWR']])
mytab[['GWR']]$config_model
mytab[['GWR']]$CV
summary(mytab[['GWR']]$model$Betav)

mytab[['GWR_2']]$config_model
mytab[['GWR_2']]$CV
```
bisq

bisq

bisquare kernel

Description

bisquare kernel

Usage

bisq(d, h)

Arguments

d  a vector of distance

h  a distance bandwidth
**Value**

a vector of weight

**Examples**

```r
w <- bisq(-30:30, 20)
plot(-30:30, w, type='l')
abline(v=-20)
abline(v=20)
```

---

**bisq_C**  
**bisquare kernel, RcppEigen version**

**Description**

bisquare kernel, RcppEigen version

**Usage**

`bisq_C(d, h, Minv)`

**Arguments**

- `d` a vector of distance
- `h` a distance bandwidth
- `Minv` Minimal number of non null weight (neigbor)

**Value**

a vector of weight

**Examples**

```r
w <- bisq_C(1:100,20,0)
plot(1:100,w,type='l')
```
bisq_knn_C

Adaptive bisquare kernel

Description
Adaptive bisquare kernel

Usage
bisq_knn_C(d, h)

Arguments
- d: a vector of distance
- h: bandwidth size expressed in number of neighbors

Value
a vector of weight

Examples
w = bisq_knn_C(-30:30, 20)
plot(-30:30, w, type='l')
abline(v=-10)
abline(v=10)

gauss_adapt
Adaptive gaussian kernel

Description
Adaptive gaussian kernel

Usage
gauss_adapt(d, h)

Arguments
- d: a vector of distance
- h: bandwidth size expressed in number of neighbors
Value

a vector of weight

Examples

```r
w = gauss_adapt_c(-30:30, 20)
plot(-30:30, w, type='l')
abline(v=-10)
abline(v=10)
```

---

gauss_adapt_C

Adaptive gaussian kernel, RcppEigen version

Description

Adaptive gaussian kernel, RcppEigen version

Usage

`gauss_adapt_c(d, h)`

Arguments

d a vector of distance

h bandwidth size expressed in number of neighbors

Value

a vector of weight

Examples

```r
w = gauss_adapt_c(-30:30, 20)
plot(-30:30, w, type='l')
abline(v=-10)
abline(v=10)
```
**Description**

`kernelW_C` is a function that computes weight matrix for a given kernel and bandwidth.

**Usage**

```r
kernelW_C(XX, hh, MykernelS, isgcv_, Type, Minv, maxknn_, NmaxDist_, TIME, Decay, DDiagNull)
```

**Arguments**

- **XX**: a matrix with coordinates.
- **hh**: a bandwidth value.
- **MykernelS**: a kernel type between ('bin', 'bisq', 'gauss', 'gauss_adapt', 'knn', 'bisq_knn').
- **isgcv_**: default FALSE for computing CV criteria (for example for selecting optimal bandwidth).
- **Type**: Kernel type.
- **Minv_**: minimum number of neighbors when using distance kernels.
- **maxknn_**: default 500, when n>NmaxDist only maxknn first neighbours are used for computation distance.
- **NmaxDist_**: default 5000, when n>NmaxDist only maxknn first neighbours are used for computing distance.
- **TIME**: default FALSE, time is used for computing weights if TIME is TRUE weights for future observation are set to zero.
- **Decay**: time decay when time is used for computing weights.
- **DDiagNull**: default FALSE, if TRUE diagonal has zero weights.

**Value**

`kernelW_C` returns a sparse weight matrix.

**See Also**

`MGWRSAR`, `bandwidths_mgwrsar`, `summary_mgwrsar`, `plot_mgwrsar`, `predict_mgwrsar`.

**Examples**

```r
data(data_mgwrsar); coord=as.matrix(mydata[,c("x_lat","y_lon")]);
W=kernelW_C(coord,100,'bisq_knn',FALSE,'GD',1,500,5000,FALSE,0,FALSE)
plot(D_dense_C(coord[1,1],coord[1,2],coord[,1],coord[,2]),W[1,])
```
**KNN**

A function that returns a row normalized weight matrix based on k first neighbors, to be documented

**Description**

A function that returns a row normalized weight matrix based on k first neighbors, to be documented

**Usage**

\[ \text{KNN}(\text{coord}, h, \text{diagnull}=\text{TRUE}, \text{kernel}='\text{knn}', \text{query}=\text{NULL}) \]

**Arguments**

- `coord`: matrix of coordinates
- `h`: a bandwidth
- `diagnull`: 0 on diagonal, default TRUE
- `kernel`: kernel type ('bisq', 'bisq_knn', 'gauss', 'gauss_adapt', 'knn')
- `query`: an index of neighbors to consider, if NULL all observation are used.

**Value**

a row nominalized weight \( \text{dgCmatrix} \)

**Examples**

```r
data(data_mgwrsar)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
W=KNN(coord,8)
which(W[1,]>0)
W[1,which(W[1,]>0)]
```

**MGWRSAR**

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

**Description**

MGWRSAR is 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**: vector containing the kernel types. Possible types: k nearest neighbors ("knn"), bisquare ("bisq"), adaptative bisquare ("bisq_knn"), gaussian ("gauss"), adaptative gaussian ("gauss_adapt").
- **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 Aurocorrelation, default NULL.
- type verbose mode, default FALSE.
- 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'.
- isgcv computing CV 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.
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.

**MGWRSAR** is a wrapper function for estimating linear and local linear model with spatial autocorrelation that allows to estimate the following models:

\[
y = \beta_v(u_i, v_i)X_v + \epsilon_i \quad \text{(MGWR)}
\]

\[
y = \lambda Wy + \beta_c X_c + \epsilon_i \quad \text{(MGWR-SAR(0,0,k))}
\]

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, kernelW_C

Examples

data(data_mgwrsar)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
model_GWR<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata,coord=coord,
fixed_vars=NULL,kernels=c('gauss_knn'),
H=128, Model = 'GWR',control=list())
summary_mgwrsar(model_GWR)

W=KNN(coord,8)
model_MGWRSAR_0_kc_kv<-MGWRSAR(formula = 'Y_mgwrsar_0_kc_kv~X1+X2+X3', data = mydata,
coord=coord, fixed_vars='Intercept', kernels=c('gauss_adapt'),
H=128, Model = 'MGWRSAR_0_kc_kv',control=list(W=W))
summary_mgwrsar(model_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

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 which residuals are simulated, 'H0' or 'H1' (default)
- df: Hypothesis under which degree of freedom is estimated.
- focal: see sample_stat help
- D: A matrix of distance

Value

The value of the statistic test and a p ratio.

See Also

mgwrsar_bootstrap_test_all

Examples

data(data_mgwrsar)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
model_GWR<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata, coord=coord,
fixed_vars=NULL,kernels=c('gauss_adapt'), H=20,
Model = 'GWR',control=list(SE=TRUE))

model_MGWR<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata, coord=coord,
fixed_vars='Intercept',kernels=c('gauss_adapt'), H=20,
A bootstrap test for testing nullity of all Betas for mgwrsar class model.

Usage

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

Arguments

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

Value

A matrix with statistical test values and p ratios

See Also

- `mgwrsar_bootstrap_test`

`mydata`

`mydata` is a simulated data set of a mgwrsar model

Description

`mydata` is a simulated data set of a mgwrsar model

Usage

```r
mydata
```

Format

An object of class `data.frame` with 1000 rows and 22 columns.
Author(s)

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

References


plot_mgwrsar

plot_mgwrsar plots the value of local parameters of a mgwrsar models using a leaflet map.

Description

plot_mgwrsar plots the value of local parameters of a mgwrsar models using a leaflet map.

Usage

plot_mgwrsar(model, type='coef', var=NULL, SP=NULL, SP_id=NULL, proj=NULL, mypalette= "RdYlGn", opacity=1, fopacity=1, radius=1500)

Arguments

model a mgwrsar model.
type default 'coef', for plotting the value of the coefficients. Local t-Student could also be plot using 't_coef'.
var Names of variable to plot.
SP A spdf object.
SP_id Id regions for spdf object.
proj A CRS projection.
mypalette A leaflet palette.
opacity Opacity of border color.
fopacity Opacity of fill color.
radius radius of circle for plot of points.

Value

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

See Also

MGWRSAR, bandwidths_mgwrsar, summary_mgwrsar, predict_mgwrsar, kernelW_C
**predict_mgwrsar**

**Examples**

```r
library(mgwrsar)
data(data_mgwrsar)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
W=KNN(coord,4)
model_GWR<--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_GWR)
plot_mgwrsar(model_GWR,type='t_coef',var='X1')
```

---

**predict_mgwrsar**  
*mgwrsar Model Predictions*

**Description**

`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**

```r
predict_mgwrsar(model, newdata, newdata_coord, W = NULL,
type = "BPN", h_w = 100, kernel_w = "knn", k_extra = 12,
kernel_extra = "sheppard")
```

**Arguments**

- `model`: a model of mgwrsar class.
- `newdata`: a matrix or data.frame of new data.
- `newdata_coord`: a matrix of new coordinates.
- `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`: bandwidth for constructing W, if W is NULL.
- `kernel_w`: kernel for constructing W, if W is NULL.
- `k_extra`: number of neighboors for local parameter extrapolation, default 12.
- `kernel_extra`: kernel for local parameter extrapolation, default sheppard kernel.

**Value**

A vector of predictions.
summary_mgwrsar

See Also
MGWRSAR, bandwidths_mgwrsar, summary_mgwrsar, plot_mgwrsar, kernelW_C

Examples

```r
library(mgwrsar)
data(data_mgwrsar)
coord<-as.matrix(mydata[,c("x_lat","y_lon"))
W=KNN(coord,2)
model_GWR_insample<-MGWRSAR(formula = 'Y_gwr~X1+X2+X3', data = mydata[1:800,],
coor=coord[1:800,], fixed_vars=NULL,kernels=c('gauss_adapt'),H=50,
Model = 'GWR',control=list())
Y_pred=predict_mgwrsar(model_GWR_insample, newdata=mydata[801:1000,],
newdata_coord=coord[801:1000,],k_extra = 8, kernel_extra = "sheppard")
head(Y_pred)
head(mydata$Y_gwr[801:1000])
sqrt(mean((mydata$Y_gwr[801:1000]-Y_pred)^2))

## predict with spatial autocorrelation
model_MGWRSAR_1_0_kv_insample<-MGWRSAR(formula = 'Y_mgwrsar_1_0_kv~X1+X2+X3',data = mydata[1:800,],
coor=coord[1:800,], fixed_vars=NULL,kernels=c('gauss_adapt'),H=50,
Model = 'MGWRSAR_1_0_kv',control=list(W=W[1:800,1:800],Lambdacor=TRUE,SE=TRUE))
summary_mgwrsar(model_MGWRSAR_1_0_kv_insample)

## with BLUP
Y_pred=predict_mgwrsar(model_MGWRSAR_1_0_kv_insample, newdata=mydata[801:1000,],
newdata_coord=coord[801:1000,], k_extra = 12, W = W,
type = "BPN",kernel_extra = "sheppard")
head(Y_pred)
head(mydata$Y_gwr[801:1000])
sqrt(mean((mydata$Y_gwr[801:1000]-Y_pred)^2))

## without BLUP
Y_pred=predict_mgwrsar(model_MGWRSAR_1_0_kv_insample, newdata=mydata[801:1000,],
newdata_coord=coord[801:1000,], k_extra = 12, W = W,
type = "TC",kernel_extra = "sheppard")
head(Y_pred)
head(mydata$Y_mgwrsar_1_0_kv[801:1000])
sqrt(mean((mydata$Y_mgwrsar_1_0_kv[801:1000]-Y_pred)^2))
```

**Description**
Print a summary of mgwrsar models
Summary

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, kernelW_C

Examples

## Not run:
library(mgwrsar)
data(data_mgwrsar)
coord=as.matrix(mydata[,c("x_lat","y_lon")])
W=KNN(coord,8)
model_GWR=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_GWR)

## End(Not run)
Index

*Topic GWR, mixed GWR, LWR, local regression, SAR, spatial autocorrelation
  mgwrsar-package, 2
*Topic datasets
  mydata, 17

bandwidths_mgwrsar, 4
bisq, 7
bisq_C, 8
bisq_knn_C, 9

gauss_adapt, 9
gauss_adapt_C, 10

kernelW_C, 11
KNN, 12

locfit, 3
lwr, 3

MGWRSAR, 12
mgwrsar (mgwrsar-package), 2
mgwrsar-package, 2
mgwrsar_bootstrap_test, 16
mgwrsar_bootstrap_test_all, 17
mydata, 17

plot_mgwrsar, 18
predict_mgwrsar, 19

summary_mgwrsar, 20