Package 'GWmodel'

February 1, 2015

Version 1.2-5

Date 2015-02-01

Title Geographically-Weighted Models

Depends R (>= 3.0.0),maptools (>= 0.5-2), robustbase,sp

Suggests mvoutlier, RColorBrewer, gstat

Description In GWmodel, we introduce techniques from a particular branch of spatial statistics,termed geographically-weighted (GW) models. GW models suit situations when data are not described well by some global model, but where there are spatial regions where a suitably localised calibration provides a better description. GWmodel includes functions to calibrate: GW summary statistics, GW principal components analysis, GW discriminant analysis and various forms of GW regression; some of which are provided in basic and robust (outlier resistant) forms.

Author Binbin Lu[aut], Paul Harris[aut], Martin Charlton[aut], Chris Brunsdon[aut], Tomoki Nakaya[aut], Isabella Gollini[ctb]

Maintainer Binbin Lu <binbinlu@whu.edu.cn>

Repository CRAN

URL http://gwr.nuim.ie/

NeedsCompilation no

Date/Publication 2015-02-01 17:38:21

R topics documented:

`GWmodel-package` *Geographically-Weighted Models*

Description

In GWmodel, we introduce techniques from a particular branch of spatial statistics, termed geographically-weighted (GW) models. GW models suit situations when data are not described well by some global model, but where there are spatial regions where a suitably localised calibration provides a better description. GWmodel includes functions to calibrate: GW summary statistics, GW principal components analysis, GW discriminant analysis and various forms of GW regression; some of which are provided in basic and robust (outlier resistant) forms.

Details

Package:	GWmodel
Type:	Package
Version:	1.2-5
Date:	2015-02-01
License:	GPL (>=2)
LazyLoad:	yes

Note

Acknowledgement: We gratefully acknowledge support from Science Foundation Ireland under the National Development Plan through the award of a Strategic Research Centre grant 07-SRC-I1168.

Author(s)

Binbin Lu, Paul Harris, Martin Charlton, Chris Brunsdon, Tomoki Nakaya, Isabella Gollini

Maintainer: Binbin Lu <binbinlu@whu.edu.cn>

References

Gollini I, Lu B, Charlton M, Brunsdon C, Harris P (2015) GWmodel: an R Package for exploring Spatial Heterogeneity using Geographically Weighted Models. Journal of Statistical Software, 63(17):1-50, http://www.jstatsoft.org/v63/i17/

Lu B, Harris P, Charlton M, Brunsdon C (2014) The GWmodel R Package: further topics for exploring Spatial Heterogeneity using Geographically Weighted Models. Geo-spatial Information Science 17(2): 85-101, http://www.tandfonline.com/doi/abs/10.1080/10095020.2014.917453

bw.ggwr	*Bandwidth selection for generalised geographically weighted regression (GWR)*

Description

A function for bandwidth selection to calibrate a generalised GWR model

Usage

```
bw.ggwr(formula, data, family ="poisson", approach="CV",
kernel="bisquare",adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`family`	a description of the error distribution and link function to be used in the model, which can be specified by “poisson” or “binomial”
`approach`	specified by CV for cross-validation approach or by AIC corrected (AICc) approach

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

Returns the adaptive or fixed distance bandwidth

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

bw.gwda *Bandwidth selection for GW Discriminant Analysis*

Description

A function for bandwidth selection for GW Discriminant Analysis

Usage

```
bw.gwda(formula, data, COV.gw = T, prior.gw = T, mean.gw = T,
                  prior = NULL, wqda = F, kernel = "bisquare", adaptive
                  = FALSE, p = 2, theta = 0, longlat = F,dMat)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame for training, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`COV.gw`	if true, localised variance-covariance matrix is used for GW discriminant analysis; otherwise, global variance-covariance matrix is used
`mean.gw`	if true, localised mean is used for GW discriminant analysis; otherwise, global mean is used
`prior.gw`	if true, localised prior probability is used for GW discriminant analysis; otherwise, fixed prior probability is used
`prior`	a vector of given prior probability
`wqda`	if TRUE, weighted quadratic discriminant analysis will be applied; otherwise weighted linear discriminant analysis will be applied

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

Returns the adaptive or fixed distance bandwidth.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Brunsdon, C, Fotheringham S, and Charlton, M (2007), Geographically Weighted Discriminant Analysis, Geographical Analysis 39: 376-396

---

bw.gwpca — *Bandwidth selection for Geographically Weighted Principal Components Analysis (GWPCA)*

Description

A function for bandwidth selection to calibrate a basic or robust GWPCA

Usage

```
bw.gwpca(data,vars,k=2, robust=FALSE,kernel="bisquare",adaptive=FALSE,p=2,
         theta=0, longlat=F,dMat)
```

Arguments

`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`vars`	a vector of variable names to be evaluated
`k`	the number of retained components, and it must be less than the number of variables
`robust`	if TRUE, robust GWPCA will be applied; otherwise basic GWPCA will be applied

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance

theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

Returns the adaptive or fixed distance bandwidth

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

bw.gwr *Bandwidth selection for basic GWR*

Description

A function for bandwidth selection to calibrate a basic GWR model

Usage

```
bw.gwr(formula, data, approach="CV",kernel="bisquare",
                 adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

formula	Regression model formula of a formula object
data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
approach	specified by CV for cross-validation approach or by AIC corrected (AICc) approach

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

Returns the adaptive or fixed distance bandwidth

Note

For a discontinuous kernel function, a bandwidth can be specified either as a fixed (constant) distance or as a fixed (constant) number of local data (i.e. an adaptive distance). For a continuous kernel function, a bandwidth can be specified either as a fixed distance or as a 'fixed quantity that reflects local sample size' (i.e. still an 'adaptive' distance but the actual local sample size will be the sample size as functions are continuous). In practise a fixed bandwidth suits fairly regular sample configurations whilst an adaptive bandwidth suits highly irregular sample configurations. Adaptive bandwidths ensure sufficient (and constant) local information for each local calibration. This note is applicable to GW regression and all other GW models

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

bw.gwr.lcr *Bandwidth selection for locally compensated ridge GWR (GWR-LCR)*

Description

This function finds an optimal bandwidth for gwr.lcr via a cross-validation approach

Usage

```
bw.gwr.lcr(formula, data, kernel="bisquare",
        lambda=0,lambda.adjust=FALSE,cn.thresh=NA,
        adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

formula	Regression model formula of a formula object
data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
lambda	option for a globally-defined (constant) ridge parameter. Default is lambda=0, which gives a basic GWR fit

`lambda.adjust`	a locally-varying ridge parameter. Default FALSE, refers to: (i) a basic GWR without a local ridge adjustment (i.e. lambda=0, everywhere); or (ii) a penalised GWR with a global ridge adjustment (i.e. lambda is user-specified as some constant, other than 0 everywhere); if TRUE, use cn.tresh to set the maximum condition number. Here for locations with a condition number (for its local design matrix) above this user-specified threshold, a local ridge parameter is found
`cn.thresh`	maximum value for condition number, commonly set between 20 and 30
`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

Returns the adaptive or fixed distance bandwidth

Author(s)

Binbin Lu `<binbinlu@whu.edu.cn>`

`check.components` *Interaction tool with the GWPCA glyph map*

Description

The function interacts with the multivariate glyph plot of GWPCA loadings.

Usage

```
check.components(ld,loc)
```

Arguments

`ld`	GWPCA loadings returned by gwpca
`loc`	a 2-column numeric array of GWPCA evaluation locations

Author(s)

Binbin Lu `<binbinlu@whu.edu.cn>`

See Also

glyph.plot

DubVoter *Voter turnout data in Greater Dublin(SpatialPolygonsDataFrame)*

Description

Voter turnout and social characters data in Greater Dublin for the 2002 General election and the 2002 census. Note that this data set was originally thought to relate to 2004, so for continuity we have retained the associated variable names.

Usage

```
data(DubVoter)
```

Format

A SpatialPolygonsDataFrame with 322 electoral divisions on the following 11 variables.

DED_ID a vector of ID

X a numeric vector of x coordinates

Y a numeric vector of y coordinates

DiffAdd percentage of the population in each ED who are one-year migrants (i.e. moved to a different address 1 year ago)

LARent percentage of the population in each ED who are local authority renters

SC1 percentage of the population in each ED who are social class one (high social class)

Unempl percentage of the population in each ED who are unemployed

LowEduc percentage of the population in each ED who are with little formal education

Age18_24 percentage of the population in each ED who are age group 18-24

Age25_44 percentage of the population in each ED who are age group 25-44

Age45_64 percentage of the population in each ED who are age group 45-64

GenEl2004 percentage of population in each ED who voted in 2004 election

Details

Variables are from DubVoter.shp.

References

Kavanagh A (2006) Turnout or turned off? Electoral participation in Dublin in the early 21st Century. Journal of Irish Urban Studies 3(2):1-24

Harris P, Brunsdon C, Charlton M (2011) Geographically weighted principal components analysis. International Journal of Geographical Information Science 25 (10):1717-1736

Examples

```
data(DubVoter)
ls()
## Not run:
spplot(Dub.voter,names(Dub.voter)[4:12])

## End(Not run)
```

EWHP *House price data set (DataFrame) in England and Wales*

Description

A house price data set over the England and Wales with 9 hedonic variables from 1999.

Usage

```
data(EWHP)
```

Format

A data frame with 519 observations on the following 12 variables.

Easting a numeric vector, X coordinate

Northing a numeric vector, Y coordinate

PurPrice a numeric vector, the purchase price of the property

BldIntWr a numeric vector, 1 if the property was built during the world war, 0 otherwise

BldPostW a numeric vector, 1 if the property was built after the world war, 0 otherwise

Bld60s a numeric vector, 1 if the property was built between 1960 and 1969, 0 otherwise

Bld70s a numeric vector, 1 if the property was built between 1970 and 1979, 0 otherwise

Bld80s a numeric vector, 1 if the property was built between 1980 and 1989, 0 otherwise

TypDetch a numeric vector, 1 if the property is detached (i.e. it is a stand-alone house), 0 otherwise

TypSemiD a numeric vector, 1 if the property is semi detached, 0 otherwise

TypFlat a numeric vector, if the property is a flat (or 'apartment' in the USA), 0 otherwise

FlrArea a numeric vector, floor area of the property in square metres

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham, A.S., Brunsdon, C., and Charlton, M.E. (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Examples

```
###
data(EWHP)
head(ewhp)
houses.spdf <- SpatialPointsDataFrame(ewhp[, 1:2], ewhp)
 ####Get the border of England and Wales
data(EWOutline)
plot(ewoutline)
plot(houses.spdf, add = TRUE, pch = 16)
```

EWOutline — *Outline of England and Wales for data EWHP*

Description

Outline (SpatialPolygonsDataFrame) of England and Wales for data EWHP.

Usage

```
data(EWOutline)
```

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

Georgia — *Georgia census data set (csv file)*

Description

The Georgia census data set from Fotheringham et al. (2002).

Usage

```
data(Georgia)
```

Format

A data frame with 159 observations on the following 13 variables.

AreaKey An identification number for each county

Latitude The latitude of the county centroid

Longitud The longitude of the county centroid

TotPop90 Population of the county in 1990

PctRural Percentage of the county population defined as rural

PctBach Percentage of the county population with a bachelors degree

PctEld Percentage of the county population aged 65 or over

PctFB Percentage of the county population born outside the US

PctPov Percentage of the county population living below the poverty line

PctBlack Percentage of the county population who are black

ID a numeric vector of IDs

X a numeric vector of x coordinates

Y a numeric vector of y coordinates

Details

This data set can also be found in GWR 3 and in spgwr.

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Examples

```
data(Georgia)
ls()
coords <- cbind(Gedu.df$X, Gedu.df$Y)
educ.spdf <- SpatialPointsDataFrame(coords, Gedu.df)
spplot(educ.spdf, names(educ.spdf)[4:10])
```

GeorgiaCounties *Georgia counties data (SpatialPolygonsDataFrame)*

Description

The Georgia counties data used for Georgia census data.

Usage

```
data(GeorgiaCounties)
```

Details

Variables are from GWR3 file GData_utm.csv.

Examples

```
data(GeorgiaCounties)
plot(Gedu.counties)
data(Georgia)
coords <- cbind(Gedu.df$X, Gedu.df$Y)
educ.spdf <- SpatialPointsDataFrame(coords, Gedu.df)
plot(educ.spdf, add=TRUE)
```

---

ggwr.cv — *Cross-validation score for a specified bandwidth for generalised GWR*

Description

This function finds the cross-validation score for a specified bandwidth for generalised GWR

Usage

```
ggwr.cv(bw, X, Y,family="poisson", kernel="bisquare",adaptive=F, dp.locat,
        p=2, theta=0, longlat=F,dMat)
```

Arguments

`bw`	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`X`	a numeric matrix of the independent data with an extra column of "ones" for the 1st column
`Y`	a column vector of the dependent data
`family`	a description of the error distribution and link function to be used in the model, which can be specified by "poisson" or "binomial"

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`dp.locat`	a two-column numeric array of observation coordinates
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

CV.score	cross-validation score

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

ggwr.cv.contrib   *Cross-validation data at each observation location for a generalised GWR model*

Description

This function finds the cross-validation data at each observation location for a generalised GWR model with a specified bandwidth. Can be used to detect outliers.

Usage

```
ggwr.cv.contrib(bw, X, Y,family="poisson", kernel="bisquare",adaptive=F,
                dp.locat, p=2, theta=0, longlat=F,dMat)
```

Arguments

bw	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
X	a numeric matrix of the independent data with an extra column of "ones" for the 1st column
Y	a column vector of the dependent data
family	a description of the error distribution and link function to be used in the model, which can be specified by "poisson" or "binomial"

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
dp.locat	a two-column numeric array of observation coordinates
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

`CV`	a data vector consisting of squared residuals, whose sum is the cross-validation score for the specified bandwidth

Author(s)

Binbin Lu <`binbinlu@whu.edu.cn`>

`glyph.plot` *Multivariate glyph plots of GWPCA loadings*

Description

This function provides a multivariate glyph plot of GWPCA loadings at each output location.

Usage

```
glyph.plot(ld,loc, r1=50, add=FALSE,alpha=1,sep.contrasts=FALSE)
```

Arguments

`ld`	GWPCA loadings returned by gwpca
`loc`	a two-column numeric array for providing evaluation locations of GWPCA calibration
`r1`	argument for the size of the glyphs, default is 50; glyphs get larger as r1 is reduced
`add`	if TRUE, add the plot to the existing window.
`alpha`	the level of transparency of glyph from function rgb() and ranges from 0 to max (fully transparent to opaque)
`sep.contrasts`	allows different types of glyphs and relates to whether absolute loadings are used (TRUE) or not

Author(s)

Binbin Lu <`binbinlu@whu.edu.cn`>

References

Harris P, Brunsdon C, Charlton M (2011) Geographically weighted principal components analysis. International Journal of Geographical Information Science 25:1717-1736

gw.dist *Distance matrix calculation*

Description

Calculate a distance matrix between any GW model calibration points and the data points.

Usage

```
gw.dist(dp.locat, rp.locat, focus=0, p=2, theta=0, longlat=F)
```

Arguments

dp.locat	a numeric matrix of two columns giving the coordinates of the data points
rp.locat	a numeric matrix of two columns giving the coordinates of the GW model calibration points
focus	an integer, indexing to the current GW model point, if focus=0, all the distances between all the GW model calibration points and data points will be calculated and a distance matrix will be returned; if 0<focus<length(rp.locat), then the distances between the 'focus'th GW model points and data points will be calculated and a distance vector will be returned
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated

Value

returns a numeric distance matrix or vector; matrix with its rows corresponding to the observations and its columns corresponds to the GW model calibration points.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

See Also

dist in **stats**

Examples

```
dp<-cbind(sample(100),sample(100))
rp<-cbind(sample(10),sample(10))
#Euclidean distance metric is used.
dist.v1<-gw.dist(dp.locat=dp, focus=5, p=2, theta=0, longlat=FALSE)
#Manhattan distance metric is used.
#The coordinate system is rotated by an angle 0.5 in radian.
dist.v2<-gw.dist(dp.locat=dp, focus=5, p=1, theta=0.5)
```

```
#Great Circle distance metric is used.
dist.v3<-gw.dist(dp.locat=dp, focus=5, longlat=TRUE)
#A generalized Minkowski distance metric is used with p= 0.75 .
#The coordinate system is rotated by an angle 0.8 in radian.
dist.v4<-gw.dist(dp.locat=dp,rp.locat=rp, focus=5, p=0.75,theta=0.8)
################################
#matrix is calculated
#Euclidean distance metric is used.
dist.m1<-gw.dist(dp.locat=dp, p=2, theta=0, longlat=FALSE)
#Manhattan distance metric is used.
#The coordinate system is rotated by an angle 0.5 in radian.
dist.m2<-gw.dist(dp.locat=dp, p=1, theta=0.5)
#Great Circle distance metric is used.
#dist.m3<-gw.dist(dp.locat=dp, longlat=TRUE)
#A generalized Minkowski distance metric is used with p= 0.75 .
#The coordinate system is rotated by an angle 0.8 in radian.
dist.m4<-gw.dist(dp.locat=dp,rp.locat=rp, p=0.75,theta=0.8)
```

---

gw.pcplot *Geographically weighted parallel coordinate plot for investigating multivariate data sets*

Description

This function provides a geographically weighted parallel coordinate plot for investigating a multivariate data set. It has an option that weights the lines of the plot with increasing levels of transparency, according to their observation's distance from a specified focal/observation point. This plot can be used to identify outliers.

Usage

```
gw.pcplot(data,vars,focus,bw,adaptive = FALSE, ylim=NULL,ylab="",fixtrans=FALSE,
          p=2, theta=0, longlat=F,dMat,...)
```

Arguments

data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
vars	a vector of variable names to be evaluated
focus	an integer, indexing to the observation point
bw	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
ylim	the y limits of the plot
ylab	a label for the y axis

`fixtrans`	if TRUE, the transparency of the neighbouring observation plot lines increases with distance; If FALSE a standard (non-spatial) parallel coordinate plot is returned.
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`...`	other graphical parameters, (see par)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Harris P, Brunsdon C, Charlton M, Juggins S, Clarke A (2014) Multivariate spatial outlier detection using robust geographically weighted methods. Mathematical Geosciences 46(1) 1-31

---

gwda *GW Discriminant Analysis*

Description

This function implements GW discriminant analysis.

Usage

```
gwda(formula, data, predict.data,validation = T, COV.gw=T,
                 mean.gw=T, prior.gw=T, prior=NULL, wqda =F,
                kernel = "bisquare", adaptive = FALSE, bw,
                 p = 2, theta = 0, longlat = F,dMat)
## S3 method for class 'gwda'
print(x, ...)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame for training, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`predict.data`	a Spatial*DataFrame object for prediction, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**; if it is not given, the traing data will be predicted using leave-one-out cross-validation.
`validation`	If TRUE, the results from the prediction will be validated and the correct proportion will be calculated.

`COV.gw`	if true, localised variance-covariance matrix is used for GW discriminant analysis; otherwise, global variance-covariance matrix is used
`mean.gw`	if true, localised mean is used for GW discriminant analysis; otherwise, global mean is used
`prior.gw`	if true, localised prior probability is used for GW discriminant analysis; otherwise, fixed prior probability is used
`prior`	a vector of given prior probability
`wqda`	if TRUE, weighted quadratic discriminant analysis will be applied; otherwise weighted linear discriminant analysis will be applied

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwpca;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`x`	an object of class "gwda"
`...`	arguments passed through (unused)

Value

A class of object "gwda"

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Brunsdon, C, Fotheringham S, and Charlton, M (2007), Geographically Weighted Discriminant Analysis, Geographical Analysis 39:376-396

gwpca *GWPCA*

Description

This function implements basic or robust GWPCA.

Usage

```
gwpca(data, elocat, vars, k = 2, robust = FALSE, kernel = "bisquare",
                  adaptive = FALSE, bw, p = 2, theta = 0, longlat = F, cv = T,
                  dMat)
```

Arguments

`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`elocat`	a two-column numeric array or Spatial*DataFrame object for providing evaluation locations, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`vars`	a vector of variable names to be evaluated
`k`	the number of retained components; k must be less than the number of variables
`robust`	if TRUE, robust GWPCA will be applied; otherwise basic GWPCA will be applied

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwpca;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`cv`	If TRUE, cross-validation data will be found that are used to calculate the cross-validation score for the specified bandwidth.
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

A list of components:

`loadings`	The coefficients of the variables for each component score
`var`	The amount of variance accounted for by each component
`GW.arguments`	A list of geographically weighted arguments supplied to the function call
`CV`	Vector of cross-validation data

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Harris P, Brunsdon C, Charlton M (2011) Geographically weighted principal components analysis. International Journal of Geographical Information Science 25:1717-1736

Harris P, Brunsdon C, Charlton M, Juggins S, Clarke A (2014) Multivariate spatial outlier detection using robust geographically weighted methods. Mathematical Geosciences 46(1) 1-31

Harris P, Clarke A, Juggins S, Brunsdon C, Charlton M (2014) Geographically weighted methods and their use in network re-designs for environmental monitoring. Stochastic Environmental Research and Risk Assessment 28: 1869-1887

Harris P, Clarke A, Juggins S, Brunsdon C, Charlton M (2015) Enhancements to a geographically weighted principal components analysis in the context of an application to an environmental data set. Geographical Analysis DOI: 10.1111/gean.12048

Examples

```
## Not run:
if(require("mvoutlier") && require("RColorBrewer"))
{
  data(bsstop)
  Data.1 <- bsstop[, 1:14]
  colnames(Data.1)
  Data.1.scaled <- scale(as.matrix(Data.1[5:14]))  # standardised data...
  rownames(Data.1.scaled) <- Data.1[, 1]
  #compute principal components:
  pca <- princomp(Data.1.scaled, cor = FALSE, scores = TRUE)
  # use covariance matrix to match the following...
  pca$loadings
  data(bss.background)
  backdrop <- function()
   plot(bss.background, asp = 1, type = "l", xaxt = "n", yaxt = "n",
   xlab = "", ylab = "", bty = "n", col = "grey")
  pc1 <- pca$scores[, 1]
  backdrop()
  points(Data.1$XCOO[pc1 > 0], Data.1$YCOO[pc1 > 0], pch = 16, col = "blue")
```

```
  points(Data.1$XCOO[pc1 < 0], Data.1$YCOO[pc1 < 0], pch = 16, col = "red")

  #Geographically Weighted PCA and mapping the local loadings
  # Coordinates of the sites
  Coords1 <- as.matrix(cbind(Data.1$XCOO,Data.1$YCOO))
  d1s <- SpatialPointsDataFrame(Coords1,as.data.frame(Data.1.scaled))
  pca.gw <- gwpca(d1s,vars=colnames(d1s@data),bw=1000000,k=10)
  local.loadings <- pca.gw$loadings[, , 1]

  # Mapping the winning variable with the highest absolute loading
  # note first component only - would need to explore all components..

  lead.item <- colnames(local.loadings)[max.col(abs(local.loadings))]
  df1p = SpatialPointsDataFrame(Coords1, data.frame(lead = lead.item))
  backdrop()
  colour <- brewer.pal(8, "Dark2")[match(df1p$lead, unique(df1p$lead))]
  plot(df1p, pch = 18, col = colour, add = TRUE)
  legend("topleft", as.character(unique(df1p$lead)), pch = 18, col =
      brewer.pal(8, "Dark2"))
  backdrop()

  #Glyph plots give a view of all the local loadings together
  glyph.plot(local.loadings, Coords1, add = TRUE)

  #it is not immediately clear how to interpret the glyphs fully,
  #so inter-actively identify the full loading information using:
  check.components(local.loadings, Coords1)

  # GWPCA with an optimal bandwidth
  bw.choice <- bw.gwpca(d1s,vars=colnames(d1s@data),k=2)
  pca.gw.auto  <- gwpca(d1s,vars=colnames(d1s@data),bw=bw.choice,k=2)
  # note first component only - would need to explore all components..
  local.loadings <- pca.gw.auto$loadings[, , 1]

  lead.item <- colnames(local.loadings)[max.col(abs(local.loadings))]
  df1p = SpatialPointsDataFrame(Coords1, data.frame(lead = lead.item))
  backdrop()
  colour <- brewer.pal(8, "Dark2")[match(df1p$lead, unique(df1p$lead))]
  plot(df1p, pch = 18, col = colour, add = TRUE)
  legend("topleft", as.character(unique(df1p$lead)), pch = 18,
  col = brewer.pal(8, "Dark2"))

  # GWPCPLOT for investigating the raw multivariate data
  gw.pcplot(d1s, vars=colnames(d1s@data),focus=359, bw = bw.choice)
}

## End(Not run)
```

---

gwpca.cv *Cross-validation score for a specified bandwidth for GWPCA*

Description

This function finds the cross-validation score for a specified bandwidth for basic or robust GWPCA

Usage

```
gwpca.cv(bw,x,loc,k=2,robust=FALSE,kernel="bisquare",adaptive=FALSE,p=2,
         theta=0, longlat=F,dMat)
```

Arguments

bw	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
x	the variable matrix
loc	a two-column numeric array of observation coordinates
k	the number of retained components; k must be less than the number of variables
robust	if TRUE, robust GWPCA will be applied; otherwise basic GWPCA will be applied

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

CV.score	cross-validation score

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

gwpca.cv.contrib *Cross-validation data at each observation location for a GWPCA*

Description

This function finds the cross-validation data at each observation location for a basic or robust GW-PCA with a specified bandwidth. Can be used to detect outliers.

Usage

```
gwpca.cv.contrib(x,loc,bw, k=2,robust=FALSE,kernel="bisquare",adaptive=FALSE,
                 p=2, theta=0, longlat=F,dMat)
```

Arguments

x	the variable matrix
loc	a two-column numeric array of observation coordinates
bw	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
k	the number of retained components; k must be less than the number of variables
robust	if TRUE, robust GWPCA will be applied; otherwise basic GWPCA will be applied

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

CV	a data vector consisting of squared residuals, whose sum is the cross-validation score for the specified bandwidth (bw) and component (k).

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

gwr.basic *Basic GWR model*

Description

This function implements basic GWR

Usage

```
gwr.basic(formula, data, regression.points, bw, kernel="bisquare",
adaptive=FALSE, p=2, theta=0, longlat=F,dMat,F123.test=F,cv=T, W.vect=NULL)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`regression.points`	a Spatial*DataFrame object, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwr;fixed (distance) or adaptive bandwidth(number of nearest neighbours)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`F123.test`	If TRUE, conduct three seperate F-tests according to Leung et al. (2000).
`cv`	if TRUE, cross-validation data will be calculated and returned in the output Spatial*DataFrame
`W.vect`	default NULL, if given it will be used to weight the distance weighting matrix

Value

A list of class "gwrm":

`GW.arguments`	a list class object including the model fitting parameters for generating the report file
`GW.diagnostic`	a list class object including the diagnostic information of the model fitting
`lm`	an object of class inheriting from "lm", see lm.
`SDF`	a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with fit.points,GWR coefficient estimates, y value,predicted values, coefficient standard errors and t-values in its "data" slot.
`timings`	starting and ending time.
`this.call`	the function call used.
`Ftest.res`	results of Leung's F tests when F123.test is TRUE.

Note

The gaussian and exponential kernel functions are continuous and valued in the interval (0,1]; while bisquare, tricube and boxcar kernel functions are discontinuous and valued in the interval [0,1]. Notably, the upper limit of the bandwidth is exactly the number of observations when adaptive kernel is used. In this function, the adaptive bandwidth will be specified as the number of observations even though a larger number is assigned. In particular, the function will be the same as a global application function when the adaptive bandwidth is equal to or larger than the number of observations for boxcar kernel function.

Author(s)

Binbin Lu `<binbinlu@whu.edu.cn>`

References

Brunsdon, C, Fotheringham, S, Charlton, M (1996), Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity. Geographical Analysis 28(4):281-298

Charlton, M, Fotheringham, S, and Brunsdon, C (2007), GWR3.0, `http://gwr.nuim.ie/`.

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Leung, Y, Mei, CL, and Zhang, WX (2000), Statistical tests for spatial nonstationarity based on the geographically weighted regression model. Environment and Planning A, 32, 9-32.

Lu, B, Charlton, M, Harris, P, Fotheringham, AS (2014) Geographically weighted regression with a non-Euclidean distance metric: a case study using hedonic house price data. International Journal of Geographical Information Science 28(4): 660-681

Examples

```
data(LondonHP)
DM<-gw.dist(dp.locat=coordinates(londonhp))
##Compare the time consumed with and without a specified distance matrix
## Not run:
system.time(gwr.res<-gwr.basic(PURCHASE~FLOORSZ, data=londonhp, bw=1000,
            kernel = "gaussian"))
system.time(DM<-gw.dist(dp.locat=coordinates(londonhp)))
system.time(gwr.res<-gwr.basic(PURCHASE~FLOORSZ, data=londonhp, bw=1000,
            kernel = "gaussian", dMat=DM))

## End(Not run)
## specify an optimum bandwidth by cross-validation appraoch
bw1<-bw.gwr(PURCHASE~FLOORSZ, data=londonhp, kernel = "gaussian",dMat=DM)
gwr.res1<-gwr.basic(PURCHASE~FLOORSZ, data=londonhp, bw=bw1,kernel = "gaussian",
                   dMat=DM)
gwr.res1
data(LondonBorough)

nsa = list("SpatialPolygonsRescale", layout.north.arrow(), offset = c(561900,200900),
scale = 500, col=1)
## Not run:
if(require("RColorBrewer"))
{
  mypalette<-brewer.pal(6,"Spectral")
  x11()
  spplot(gwr.res1$SDF, "FLOORSZ", key.space = "right", cex=1.5, cuts=10,
  ylim=c(155840.8,200933.9), xlim=c(503568.2,561957.5),
  main="GWR estimated coefficients for FLOORSZ with a fixed bandwidth",
  col.regions=mypalette, sp.layout=list(nsa, londonborough))}

## End(Not run)
## Not run:
bw2<-bw.gwr(PURCHASE~FLOORSZ,approach="aic",adaptive=TRUE, data=londonhp,
            kernel = "gaussian", dMat=DM)
gwr.res2<-gwr.basic(PURCHASE~FLOORSZ, data=londonhp, bw=bw2,adaptive=TRUE,
                    kernel = "gaussian", dMat=DM)
gwr.res2
if(require("RColorBrewer"))
{
  x11()
  spplot(gwr.res2$SDF, "FLOORSZ", key.space = "right", cex=1.5, cuts=10,
  ylim=c(155840.8,200933.9), xlim=c(503568.2,561957.5),
  main="GWR estimated coefficients for FLOORSZ with an adaptive bandwidth",
  col.regions=mypalette, sp.layout=list(nsa,londonborough))}

## End(Not run)
```

`gwr.collin.diagno` *Local collinearity diagnostics for basic GWR*

Description

This function provides a series of local collinearity diagnostics for the independent variables of a basic GWR model.

Usage

```
gwr.collin.diagno(formula, data, bw, kernel="bisquare",
                   adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`bw`	bandwidth used in the weighting function, probably calculated by `bw.gwr` or `bw.gwr.lcr`; fixed (distance) or adaptive bandwidth (number of nearest neighbours)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

`corr.mat`	Local correlation matrix
`VIF`	Local Variance inflation factors (VIFs) matrix
`local_CN`	Local condition numbers
`VDP`	Local variance-decomposition proportions
`SDF`	a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with VIF, local_CN, VDP and corr.mat

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Wheeler D, Tiefelsdorf M (2005) Multicollinearity and correlation among local regression coefficients in geographically weighted regression. Journal of Geographical Systems 7:161-187

Wheeler D (2007) Diagnostic tools and a remedial method for collinearity in geographically weighted regression. Environment and Planning A 39:2464-2481

`gwr.cv` *Cross-validation score for a specified bandwidth for basic GWR*

Description

This function finds the cross-validation score for a specified bandwidth for basic GWR

Usage

```
gwr.cv(bw, X, Y, kernel="bisquare",adaptive=FALSE, dp.locat, p=2, theta=0,
       longlat=F,dMat, verbose=T)
```

Arguments

`bw`	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`X`	a numeric matrix of the independent data with an extra column of "ones" for the 1st column
`Y`	a column vector of the dependent data

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`dp.locat`	a two-column numeric array of observation coordinates
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`verbose`	if TRUE (default), reports the progress of search for bandwidth

Value

`CV.score`	cross-validation score

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

`gwr.cv.contrib` *Cross-validation data at each observation location for a basic GWR model*

Description

This function finds the cross-validation data at each observation location for a basic GWR model with a specified bandwidth. Can be used to detect outliers.

Usage

```
gwr.cv.contrib(bw, X, Y, kernel="bisquare",adaptive=FALSE, dp.locat, p=2,
               theta=0, longlat=F,dMat)
```

Arguments

`bw`	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`X`	a numeric matrix of the independent data with an extra column of "ones" for the 1st column
`Y`	a column vector of the dependent data

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`dp.locat`	a two-column numeric array of observation coordinates
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

CV	a data vector consisting of squared residuals, whose sum is the cross-validation score for the specified bandwidth.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

`gwr.generalised` *Generalised GWR models, including Poisson and Binomial options*

Description

This function implements generalised GWR

Usage

```
gwr.generalised(formula, data, regression.points, bw, family ="poisson",
 kernel="bisquare",adaptive=FALSE, p=2, theta=0, longlat=F, dMat, cv=T,tol=1.0e-5,
 maxiter=20)
```

Arguments

formula	Regression model formula of a formula object
data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
regression.points	a Spatial*DataFrame object, i.e. SpatialPointsDataFrame or SpatialPolygons-DataFrame as defined in package **sp**
bw	bandwidth used in the weighting function, possibly calculated by bw.ggwr();fixed (distance) or adaptive bandwidth(number of nearest neighbours)
family	a description of the error distribution and link function to be used in the model, which can be specified by “poisson” or “binomial”

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0

`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`cv`	if TRUE, cross-validation data will be calculated
`tol`	the threshold that determines the convergence of the IRLS procedure
`maxiter`	the maximum number of times to try the IRLS procedure

Value

A list of class "ggwrm":

`GW.arguments`	a list class object including the model fitting parameters for generating the report file
`GW.diagnostic`	a list class object including the diagnostic information of the model fitting
`glm.res`	an object of class inheriting from "glm" which inherits from the class "lm", see glm.
`SDF`	a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with fit.points,GWR coefficient estimates, y value,predicted values, coefficient standard errors and t-values in its "data" slot.
`CV`	a data vector consisting of the cross-validation data

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Charlton, M, Fotheringham, S, and Brunsdon, C (2007), GWR3.0, http://gwr.nuim.ie/.

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Examples

```
data(LondonHP)
## Not run:
DM<-gw.dist(dp.locat=coordinates(londonhp))
bw.f1 <- bw.ggwr(BATH2~FLOORSZ,data=londonhp, dMat=DM)
res.poisson<-gwr.generalised(BATH2~FLOORSZ, bw=bw.f1,data=londonhp, dMat=DM)
bw.f2 <- bw.ggwr(BATH2~FLOORSZ,data=londonhp, dMat=DM,family ="binomial")
res.binomial<-gwr.generalised(BATH2~FLOORSZ, bw=bw.f2,data=londonhp, dMat=DM,
              family ="binomial")

## End(Not run)
```

gwr.hetero *Heteroskedastic GWR*

Description

This function implements a heteroskedastic GWR model as described in Fotheringham et al. (2002, p.80-82). Related heteroskedastic GWR models can be found in Harris et al. (2010; 2011).

Usage

```
gwr.hetero(formula, data, regression.points, bw, kernel="bisquare",
                    adaptive=FALSE, tol=0.0001,maxiter=50,verbose=T,
                    p=2, theta=0, longlat=F,dMat)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`regression.points`	a Spatial*DataFrame object, i.e. SpatialPointsDataFrame or SpatialPolygons-DataFrame as defined in package **sp**
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwr;fixed (distance) or adaptive bandwidth(number of nearest neighbours)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`tol`	the threshold that determines the convergence of the iterative procedure
`maxiter`	the maximum number of times to try the iterative procedure
`verbose`	logical, if TRUE verbose output will be made from the iterative procedure
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

SDF a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with coefficient estimates in its "data" slot.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Harris P, Fotheringham AS, Juggins S (2010) Robust geographically weighed regression: a technique for quantifying spatial relationships between freshwater acidification critical loads and catchment attributes. Annals of the Association of American Geographers 100(2): 286-306

Harris P, Brunsdon C, Fotheringham AS (2011) Links, comparisons and extensions of the geographically weighted regression model when used as a spatial predictor. Stochastic Environmental Research and Risk Assessment 25:123-138

---

gwr.lcr *GWR with a locally-compensated ridge term*

Description

To address possible local collinearity problems in basic GWR, GWR-LCR finds local ridge parameters at affected locations (set by a user-specified threshold for the design matrix condition number).

Usage

```
gwr.lcr(formula, data, regression.points, bw, kernel="bisquare",
                    lambda=0,lambda.adjust=FALSE,cn.thresh=NA,
                    adaptive=FALSE, p=2, theta=0, longlat=F,cv=T,dMat)
```

Arguments

formula Regression model formula of a formula object

data a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**

regression.points
a Spatial*DataFrame object, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**, or a two-column numeric array

bw bandwidth used in the weighting function, possibly calculated by `bw.gwr.lcr`; fixed (distance) or adaptive bandwidth(number of nearest neighbours) `kernel` function chosen as follows:

gaussian: wgt = exp(-.5*(vdist/bw)^2);

exponential: wgt = exp(-vdist/bw);

bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;

tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;

boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`p` the power of the Minkowski distance, default is 2, i.e. the Euclidean distance

`lambda` option for a globally-defined (constant) ridge parameter. Default is lambda=0, which gives a basic GWR fit

`lambda.adjust` a locally-varying ridge parameter. Default FALSE, refers to: (i) a basic GWR without a local ridge adjustment (i.e. lambda=0, everywhere); or (ii) a penalised GWR with a global ridge adjustment (i.e. lambda is user-specified as some constant, other than 0 everywhere); if TRUE, use cn.tresh to set the maximum condition number. Here for locations with a condition number (for its local design matrix) above this user-specified threshold, a local ridge parameter is found

`cn.thresh` maximum value for condition number, commonly set between 20 and 30

`adaptive` if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)

`theta` an angle in radians to rotate the coordinate system, default is 0

`longlat` if TRUE, great circle distances will be calculated

`cv` if TRUE, 'cross-validation data will be calculated and returned in the output Spatial*DataFrame

`dMat` a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

A list of class "rgwr":

`SDF` a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") with coordinates of regression.points in its "data" slot.

`GW.arguments` parameters used for the LCR-GWR calibration

`GW.diagnostic` diagnostic information is given when data points are also used as regression locations

`timings` timing information for running this function

`this.call` the function call used.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Wheeler D (2007) Diagnostic tools and a remedial method for collinearity in geographically weighted regression. Environment and Planning A 39:2464-2481

Brunsdon C, Charlton M, Harris P (2012) Living with collinearity in Local Regression Models. GISRUK 2012, Lancaster, UK

Brunsdon C, Charlton M, Harris P (2012) Living with collinearity in Local Regression Models. Spatial Accuracy 2012, Brazil

Gollini I, Lu B, Charlton M, Brunsdon C, Harris P (2015) GWmodel: an R Package for exploring Spatial Heterogeneity using Geographically Weighted Models. Journal of Statistical Software 63(17): 1-50

Examples

```
data(DubVoter)
require(RColorBrewer)

# Function to find the global condition number (CN)
BKW_cn <- function (X) {
  p <- dim(X)[2]
  Xscale <- sweep(X, 2, sqrt(colSums(X^2)), "/")
  Xsvd <- svd(Xscale)$d
  cn <- Xsvd[1] / Xsvd[p]
  cn
}
#
X <- cbind(1,Dub.voter@data[,3:10])
head(X)
CN.global <- BKW_cn(X)
CN.global
## Not run:
# gwr.lcr function with a global bandwidth to check that the global CN is found
gwr.lcr1 <- gwr.lcr(GenEl2004~DiffAdd+LARent+SC1+Unempl+LowEduc+Age18_24
+Age25_44+Age45_64, data=Dub.voter, bw=10000000000)
summary(gwr.lcr1$SDF$Local_CN)

# Find and map the local CNs from a basic GWR fit using the lcr-gwr function
#(note this is NOT the locally-compensated ridge GWR fit as would need to set
#lambda.adjust=TRUE and cn.thresh=30, say)

bw.lcr2 <- bw.gwr.lcr(GenEl2004~DiffAdd+LARent+SC1+Unempl+LowEduc+Age18_24
+Age25_44+Age45_64, data=Dub.voter, kernel="bisquare", adaptive=TRUE)
gwr.lcr2 <- gwr.lcr(GenEl2004~DiffAdd+LARent+SC1+Unempl+LowEduc+Age18_24
+Age25_44+Age45_64, data=Dub.voter, bw=bw.lcr2, kernel="bisquare", adaptive=TRUE)
if(require("RColorBrewer"))
  spplot(gwr.lcr2$SDF,"Local_CN",col.regions=brewer.pal(9,"YlOrRd"),cuts=8,
  main="Local CN")

## End(Not run)
```

gwr.lcr.cv *Cross-validation score for a specified bandwidth for GWR-LCR model*

Description

This function finds the cross-validation score for a specified bandwidth for GWR-LCR

Usage

```
gwr.lcr.cv(bw,X,Y,locs,kernel="bisquare",
                    lambda=0,lambda.adjust=FALSE,cn.thresh=NA,
                    adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

bw	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
X	a numeric matrix of the independent data with an extra column of "ones" for the 1st column
Y	a column vector of the dependent data

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

locs	a two-column numeric array of observation coordinates
lambda	option for a globally-defined (constant) ridge parameter. Default is lambda=0, which gives a basic GWR fit
lambda.adjust	a locally-varying ridge parameter. Default FALSE, refers to: (i) a basic GWR without a local ridge adjustment (i.e. lambda=0, everywhere); or (ii) a penalised GWR with a global ridge adjustment (i.e. lambda is user-specified as some constant, other than 0 everywhere); if TRUE, use cn.tresh to set the maximum condition number. Here for locations with a condition number (for its local design matrix) above this user-specified threshold, a local ridge parameter is found
cn.thresh	maximum value for condition number, commonly set between 20 and 30
adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

`CV.score`	cross-validation score

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

`gwr.lcr.cv.contrib` *Cross-validation data at each observation location for the GWR-LCR model*

Description

This function finds the cross-validation data at each observation location for a GWR-LCR model with a specified bandwidth. Can be used to detect outliers.

Usage

```
gwr.lcr.cv.contrib(bw,X,Y,locs,kernel="bisquare",
                   lambda=0,lambda.adjust=FALSE,cn.thresh=NA,
                   adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

`bw`	bandwidth used in the weighting function;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`X`	a numeric matrix of the independent data with an extra column of "ones" for the 1st column
`Y`	a column vector of the dependent data
`locs`	a two-column numeric array of observation coordinates

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`lambda`	option for a globally-defined (constant) ridge parameter. Default is lambda=0, which gives a basic GWR fit
`lambda.adjust`	a locally-varying ridge parameter. Default FALSE, refers to: (i) a basic GWR without a local ridge adjustment (i.e. lambda=0, everywhere); or (ii) a penalised GWR with a global ridge adjustment (i.e. lambda is user-specified as some constant, other than 0 everywhere); if TRUE, use cn.tresh to set the maximum condition number. Here for locations with a condition number (for its local design matrix) above this user-specified threshold, a local ridge parameter is found

cn.thresh	maximum value for condition number, commonly set between 20 and 30
adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

CV	a data vector consisting of squared residuals, whose sum is the cross-validation score for the specified bandwidth.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

gwr.mixed *Mixed GWR*

Description

This function implements mixed GWR

Usage

```
gwr.mixed(formula, data, regression.points, fixed.vars,
                         intercept.fixed=FALSE, bw, diagnostic=T, kernel="bisquare",
                          adaptive=FALSE, p=2, theta=0, longlat=F,dMat)
```

Arguments

formula	Regression model formula of a formula object
data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
regression.points	a Spatial*DataFrame object, i.e. SpatialPointsDataFrame or SpatialPolygons-DataFrame as defined in package **sp**
fixed.vars	independent variables that appeared in the formula that are to be treated as global
intercept.fixed	logical, if TRUE the intercept will be treated as global
bw	bandwidth used in the weighting function, possibly calculated by bw.gwr;fixed (distance) or adaptive bandwidth(number of nearest neighbours)

`diagnostic`	logical, if TRUE the diagnostics will be calculated

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

A list of class “mgwr”:

`GW.arguments`	a list class object including the model fitting parameters for generating the report file
`aic`	AICc value from this calibration
`df.used`	effective degree of freedom
`rss`	residual sum of squares
`SDF`	a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package “sp”) integrated with coefficient estimates in its "data" slot.
`timings`	starting and ending time.
`this.call`	the function call used.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Brunsdon C, Fotheringham AS, Charlton ME (1999) Some notes on parametric signficance tests for geographically weighted regression. Journal of Regional Science 39(3):497-524

Mei L-M, He S-Y, Fang K-T (2004) A note on the mixed geographically weighted regression model. Journal of regional science 44(1):143-157

Mei L-M, Wang N, Zhang W-X (2006) Testing the importance of the explanatory variables in a mixed geographically weighted regression model. Environment and Planning A 38:587-598

`gwr.predict` *GWR used as a spatial predictor*

Description

This function implements basic GWR as a spatial predictor. The GWR prediction function is able to do leave-out-one predictions (when the observation locations are used for prediction) and predictions at a set-aside data set(when the new locations are used for prediction). It is also able to reproduce the global OLS regression prediction results.

Usage

```
gwr.predict(formula, data, predictdata, bw, kernel="bisquare",adaptive=FALSE, p=2,
          theta=0, longlat=F,dMat1, dMat2)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`predictdata`	a Spatial*DataFrame object to provide prediction locations, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwr;fixed (distance) or adaptive bandwidth(number of nearest neighbours)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat1`	a pre-specified distance matrix between data points and prediction locations; if not given, it will be calculated by the given parameters
`dMat2`	a pre-specified sysmetric distance matrix between data points; if not given, it will be calculated by the given parameters

Value

A list of class “gwrm.pred”:

- `GW.arguments` a list of geographically weighted arguments
- `SDF` a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") with GWR coefficients, predictions and prediction variances in its "data" slot.
- `this.call` the function call used.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Harris P, Fotheringham AS, Crespo R, Charlton M (2010) The use of geographically weighted regression for spatial prediction: an evaluation of models using simulated data sets. Mathematical Geosciences 42:657-680

Harris P, Juggins S (2011) Estimating freshwater critical load exceedance data for Great Britain using space-varying relationship models. Mathematical Geosciences 43: 265-292

Harris P, Brunsdon C, Fotheringham AS (2011) Links, comparisons and extensions of the geographically weighted regression model when used as a spatial predictor. Stochastic Environmental Research and Risk Assessment 25:123-138

Examples

```
## Not run:
data(LondonHP)
gwr.pred<-gwr.predict(PURCHASE~FLOORSZ, data=londonhp, bw=2000,kernel = "gaussian")
gwr.pred
#########Global regression  Compare with gstat
if(require("gstat"))
{
  mlr.g <- gstat(id = "xx1", formula = PURCHASE~FLOORSZ,data=londonhp)
  mlr.g1 <- predict(mlr.g, newdata = londonhp, BLUE = TRUE)
  mlr.g1
}
############
ols.pred<-gwr.predict(PURCHASE~FLOORSZ, data=londonhp, bw=100000000000000000000000)
ols.pred$SDF

## End(Not run)
```

gwr.robust *Robust GWR model*

Description

This function implements the two robust GWR models, as proposed in Fotheringham et al. (2002, p.73-80).

Usage

```
gwr.robust(formula, data, regression.points, bw,filtered=FALSE,
           kernel = "bisquare",adaptive = FALSE, p = 2,
           theta = 0, longlat = F, dMat, F123.test = F,
           maxiter=20,cut.filter= 3, cut1=2,cut2=3,delta=1.0e-5)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`regression.points`	a Spatial*DataFrame object, i.e. SpatialPointsDataFrame or SpatialPolygons-DataFrame as defined in package **sp**
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwr;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`filtered`	default FALSE, the automatic approach is used, if TRUE the filtered data approach is employed, as that described in Fotheringham et al. (2002 p.73-80)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`F123.test`	default FALSE, otherwise calculate F-test results (Leung et al. 2000)
`maxiter`	default 20, maximum number of iterations for the automatic approach

`cut.filter`	If filtered is TRUE, it will be used as the residual cutoff for filtering data; default cutoff is 3
`cut1`	default 2, first cutoff for the residual weighting function. wr(e)=1 if |e|<=cut1*sigma
`cut2`	default 3, second cutoff for the residual weighting function. wr(e)=(1-(|e|-2)^2)^2 if cut1*sigma<|e|<cut2*sigma, and wr(e)=0 if |e|>=cut2*sigma; cut 1 and cut2 refer to the automatic approach
`delta`	default 1.0e-5, tolerance of the iterative algorithm

Value

A list of class "gwrm":

`GW.arguments`	a list class object including the model fitting parameters for generating the report file
`GW.diagnostic`	a list class object including the diagnostic information of the model fitting
`lm`	an object of class inheriting from "lm", see lm.
`SDF`	a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with fit.points,GWR coefficient estimates, y value,predicted values, coefficient standard errors and t-values in its "data" slot. Notably, E_weigts will be also included in the output SDF which represents the residual weighting when automatic approach is used; When the filtered approach is used, E_weight is a vector consisted of 0 and 1, where 0 means outlier to be excluded from calibration.
`timings`	starting and ending time.
`this.call`	the function call used.
`Ftest.res`	results of Leung's F tests when F123.test is TRUE.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Harris P, Fotheringham AS, Juggins S (2010) Robust geographically weighed regression: a technique for quantifying spatial relationships between freshwater acidification critical loads and catchment attributes. Annals of the Association of American Geographers 100(2): 286-306

Examples

```
## Not run:
data(DubVoter)
bw.a <- bw.gwr(GenEl2004~DiffAdd+LARent+SC1+Unempl+LowEduc+Age18_24
+Age25_44+Age45_64,
data=Dub.voter,approach="AICc",kernel="bisquare",adaptive=TRUE)
bw.a
```

```
gwr.res <- gwr.basic(GenEl2004~DiffAdd+LARent+SC1+Unempl+LowEduc+Age18_24
+Age25_44+Age45_64,
data=Dub.voter,bw=bw.a,kernel="bisquare",adaptive=TRUE,F123.test=TRUE)
print(gwr.res)

# Map of the estimated coefficients for LowEduc
names(gwr.res$SDF)
if(require("RColorBrewer"))
{
  mypalette<-brewer.pal(6,"Spectral")
  X11(width=10,height=12)
  spplot(gwr.res$SDF,"LowEduc",key.space = "right",
  col.regions=mypalette,at=c(-8,-6,-4,-2,0,2,4),
  main="Basic GW regression coefficient estimates for LowEduc")
}
# Robust GW regression and map of the estimated coefficients for LowEduc
rgwr.res <- gwr.robust(GenEl2004~DiffAdd+LARent+SC1+Unempl+LowEduc+Age18_24
+Age25_44+Age45_64, data=Dub.voter,bw=bw.a,kernel="bisquare",
adaptive=TRUE,F123.test=TRUE)
print(rgwr.res)
if(require("RColorBrewer"))
{
  X11(width=10,height=12)
  spplot(rgwr.res$SDF, "LowEduc", key.space = "right",
  col.regions=mypalette,at=c(-8,-6,-4,-2,0,2,4),
  main="Robust GW regression coefficient estimates for LowEduc")
}

## End(Not run)
```

`gwr.t.adjust` *Adjust p-values for multiple hypothesis tests in basic GWR*

Description

Given a set of p-values from the pseudo t-tests of GWR outputs, this function returns adjusted p-values using: (a) Bonferroni, (b) Benjamini-Hochberg, (c) Benjamini-Yekutieli and (d) Fotheringham-Byrne procedures.

Usage

```
gwr.t.adjust(gwm.Obj)
```

Arguments

`gwm.Obj`	an object of class “gwrm”, returned by the function gwr.basic

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Byrne, G., Charlton, M. and Fotheringham, S., 2009. Multiple dependent hypothesis tests in geographically weighted regression. In: Lees, B. and Laffan, S. eds. 10th International conference on geocomputation. Sydney.

---

gwss *Geographically weighted summary statistics (GWSS)*

Description

This function calculates basic and robust GWSS. This includes geographically weighted means, standard deviations and skew. Robust alternatives include geographically weighted medians, interquartile ranges and quantile imbalances. This function also calculates basic geographically weighted covariances together with basic and robust geographically weighted correlations.

Usage

```
gwss(data, summary.locat,vars,kernel="bisquare",adaptive=FALSE, bw,p=2,
            theta=0, longlat=F,dMat,quantile=FALSE)
```

Arguments

data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
summary.locat	a Spatial*DataFrame object for providing summary locations, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
vars	a vector of variable names to be summarized
bw	bandwidth used in the weighting function

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist
quantile	if TRUE, median, interquartile range, quantile imbalance will be calculated

Value

A list of class "lss":

SDF a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") with local means,local standard deviations,local variance, local skew,local coefficients of variation, local covariances, local correlations (Pearson's), local correlations (Spearman's), local medians, local interquartile ranges, local quantile imbalances and coordinates.

... other information for reporting

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Brunsdon C, Fotheringham AS, Charlton ME (2002) Geographically weighted summary statistics - a framework for localised exploratory data analysis. Computers, Environment and Urban Systems 26:501-524

Examples

```
## Not run:
data(EWHP)
data(EWOutline)
head(ewhp)
houses.spdf <- SpatialPointsDataFrame(ewhp[, 1:2], ewhp)
localstats1 <- gwss(houses.spdf, vars = c("PurPrice", "FlrArea"), bw = 50000)
head(data.frame(localstats1$SDF))
localstats1
##A function for mapping data
if(require("RColorBrewer"))
{
   quick.map <- function(spdf,var,legend.title,main.title)
   {
     x <- spdf@data[,var]
     cut.vals <- pretty(x)
     x.cut <- cut(x,cut.vals)
     cut.levels <- levels(x.cut)
     cut.band <- match(x.cut,cut.levels)
     colors <- brewer.pal(length(cut.levels), "YlOrRd")
     colors <- rev(colors)
     par(mar=c(1,1,1,1))
     plot(ewoutline,col="olivedrab",bg="lightblue1")
     title(main.title)
     plot(spdf,add=TRUE,col=colors[cut.band],pch=16)
     legend("topleft",cut.levels,col=colors,pch=16,bty="n",title=legend.title)
  }
  quick.map(localstats1$SDF, "PurPrice_LM", "1000's Uk Pounds",
```

```
    "Geographically Weighted Mean")
    par(mfrow = c(1, 2))
    quick.map(localstats1$SDF, "PurPrice_LSKe", "Skewness Level", "Local Skewness")
   quick.map(localstats1$SDF, "PurPrice_LSD", "1000's Pounds", "Local Standard Deviation")
    #Exploring Non-Stationarity of Relationships
    quick.map(localstats1$SDF, "Corr_PurPrice.FlrArea", expression(rho),
    "Geographically Weighted Pearson Correlation")
    #Robust, Quantile Based Local Summary Statistics
    localstats2 <- gwss(houses.spdf, vars = c("PurPrice", "FlrArea"),
    bw = 50000, quantile = TRUE)
    quick.map(localstats2$SDF, "PurPrice_Median", "1000 UK Pounds",
    "Geographically Weighted Median House Price")
  }

## End(Not run)
```

LondonBorough *London boroughs data*

Description

Outline (SpatialPolygonsDataFrame) of London boroughs for the LondonHP data.

Usage

```
data(LondonBorough)
```

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

LondonHP *London house price data set (SpatialPointsDataFrame)*

Description

A house price data set with 18 hedonic variables for London in 2001 (along the river Thames area).

Usage

```
data(LondonHP)
```

Format

A SpatialPointsDataFrame object (proj4string set to "+init=epsg:27700 +datum=OSGB36").

The "data" slot is a data frame with 372 observations on the following 21 variables.

X a numeric vector, X coordinate

Y a numeric vector, Y coordinate

PURCHASE a numeric vector, the purchase price of the property

FLOORSZ a numeric vector, floor area of the property in square metres

TYPEDETCH a numeric vector, 1 if the property is detached (i.e. it is a stand-alone house), 0 otherwise

TPSEMIDTCH a numeric vector, 1 if the property is semi detached, 0 otherwise

TYPETRRD a numeric vector, 1 if the property is in a terrace of similar houses (commonly referred to as a 'row house' in the USA), 0 otherwise

TYPEBNGLW a numeric vector, if the property is a bungalow (i.e. it has only one floor), 0 otherwise

TYPEFLAT a numeric vector, if the property is a flat (or 'apartment' in the USA), 0 otherwise

BLDPWW1 a numeric vector, 1 if the property was built prior to 1914, 0 otherwise

BLDPOSTW a numeric vector, 1 if the property was built between 1940 and 1959, 0 otherwise

BLD60S a numeric vector, 1 if the property was built between 1960 and 1969, 0 otherwise

BLD70S a numeric vector, 1 if the property was built between 1970 and 1979, 0 otherwise

BLD80S a numeric vector, 1 if the property was built between 1980 and 1989, 0 otherwise

BLD90S a numeric vector, 1 if the property was built between 1990 and 2000, 0 otherwise

BATH2 a numeric vector, 1 if the property has more than 2 bathrooms, 0 otherwise

GARAGE a numeric vector,1 if the house has a garage, 0 otherwise

CENTHEAT a numeric vector, 1 if the house has central heating, 0 otherwise

BEDS2 a numeric vector, 1 if the property has more than 2 bedrooms, 0 otherwise

UNEMPLOY a numeric vector, the rate of unemployment in the census ward in which the house is located

PROF a numeric vector, the proportion of the workforce in professional or managerial occupations in the census ward in which the house is located

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham, A.S., Brunsdon, C., and Charlton, M.E. (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Lu, B, Charlton, M, Harris, P, Fotheringham, AS (2014) Geographically weighted regression with a non-Euclidean distance metric: a case study using hedonic house price data. International Journal of Geographical Information Science 28(4): 660-681

Examples

```
data(LondonHP)
data(LondonBorough)
ls()
plot(londonborough)
plot(londonhp, add=TRUE)
```

`mink.approach` *Minkovski approach*

Description

This function implements the Minkovski approach to select an 'optimum' distance metric for calibrating a GWR model.

Usage

```
mink.approach(formula, data, criterion="AIC", bw, bw.sel.approach = "AIC",adaptive=F,
          kernel="bisquare", p.vals=seq(from=0.25, to=8, length.out=32), p.inf = T,
                   theta.vals = seq(from=0, to=0.5*pi, length.out=10), verbose=F,
                        nlower = 10)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`criterion`	the criterion used for distance metric selection, AICc ("AICc") or cross-validation ("CV") score; default is "AICc"
`bw`	bandwidth used in the weighting function, possibly calculated by bw.gwr;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
`bw.sel.approach`	approach used to seclect an optimum bandwidth for each calibration if no bandwidth (bw) is given; specified by CV for cross-validation approach or by AIC corrected (AICc) approach
`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`p.vals`	a collection of positive numbers used as the power of the Minkowski distance
`p.inf`	if TRUE, Chebyshev distance is tried for model calibration, i.e. p is infinity
`theta.vals`	a collection of values used as angles in radians to rotate the coordinate system
`verbose`	if TRUE and bandwidth selection is undertaken, the bandwidth searches are reported
`nlower`	the minmum number of nearest neighbours if an adaptive kernel is used

Value

A list of:

`diag.df`	a data frame with four columns (p, theta, bandwidth, AICc/CV), each row corresponds to a calibration
`coefs.all`	a list class object including all the estimated coefficients

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Lu, B., et al. (2011). Distance metric selection for calibrating a geographically weighted regression model. The 11th International Conference on GeoComputation. London.

---

mink.matrixview *Visualisation of the results from* `mink.approach`

Description

This function visualises the AICc/CV results from the `mink.approach`.

Usage

```
mink.matrixview(diag.df, znm=colnames(diag.df)[4], criterion="AIC")
```

Arguments

`diag.df`	the first part of a list object returned by `mink.approach`
`znm`	the name of the forth column in diag.df
`criterion`	the criterion used for distance metric selection in `mink.approach`

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

`model.selection.gwr` *Model selection for GWR with a given set of independent variables*

Description

This function selects one GWR model from many alternatives based on the AICc values.

Usage

```
model.selection.gwr(DeVar=NULL,InDeVars=NULL, data=list(),bw=NULL,approach="CV",
                    adaptive=F,kernel="bisquare",dMat=NULL,p=2, theta=0, longlat=F)
```

Arguments

`DeVar`	dependent variable
`InDeVars`	a vector of independent variables for model selection
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`bw`	bandwidth used in the weighting function, possibly calculated by `bw.gwr`
`approach`	specified by **CV** (**cv**) for cross validation approach or **AIC** (**aic**) for selecting bandwidth by AICc values
`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated

Value

A list of:

`model.list`	a list of all the tried GWR models consisted of formulas and variables.
`GWR.df`	a data frame consited of four columns: bandwidth, AIC, AICc, RSS

Note

The algorithm for selecting GWR models consists of the following four steps:

Step 1. Start by calibrating all the possible bivariate GWR models by sequentially regressing a single independent variable against the dependent variable;

Step 2. Find the best performing model which produces the minimum AICc value, and permanently include the corresponding independent variable in subsequent models;

Step 3. Sequentially introduce a variable from the remaining group of independent variables to construct new models with the permanently included independent variables, and determine the next permanently included variable from the best fitting model that has the minimum AICc value;

Step 4. Repeat step 3 until all the independent variables are permanently included in the model.

In this procedure, the independent variables are iteratively included into the model in a "forward" direction. Note that there is a clear distinction between the different number of involved variables in a selection step, which can be called model levels.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Lu, B, Charlton, M, Harris, P, Fotheringham, AS (2014) Geographically weighted regression with a non-Euclidean distance metric: a case study using hedonic house price data. International Journal of Geographical Information Science 28(4): 660-681

See Also

`model.view.gwr`, `model.sort.gwr`

---

model.sort.gwr — *Sort the results of the GWR model selection function* `model.selection.gwr`.

Description

Sort the results of the GWR model seclection function `model.selection.gwr`

Usage

```
model.sort.gwr(Sorting.list , numVars, ruler.vector)
```

Arguments

`Sorting.list`	a list returned by function `model.selection.gwr`
`numVars`	the number of independent variables involved in model selection
`ruler.vector`	a numeric vector as the sorting basis

Note

The function sorts the results of model selection within invidual levels.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

See Also

model.selection.gwr, model.view.gwr

---

model.view.gwr *Visualise the GWR models from* model.selection.gwr

---

Description

This function visualises the GWR models from model.selection.gwr.

Usage

```
model.view.gwr(DeVar, InDeVars, model.list)
```

Arguments

DeVar	dependent variable
InDeVars	a vector of independent variables for model selection
model.list	a list of all GWR model tried in model.selection.gwr

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

See Also

model.selection.gwr, model.sort.gwr

Examples

```
data(LondonHP)
DM<-gw.dist(dp.locat=coordinates(londonhp))
DeVar<-"PURCHASE"
InDeVars<-c("FLOORSZ","GARAGE1","BLDPWW1","BLDPOSTW")
model.sel<-model.selection.gwr(DeVar,InDeVars, data=londonhp,
kernel = "gaussian", dMat=DM,bw=5000)
model.list<-model.sel[[1]]
model.view.gwr(DeVar, InDeVars, model.list=model.list)
```

montecarlo.gwpca.1 *Monte Carlo (randomisation) test for significance of GW PCA eigenvalue variability for the first component only - option 1*

Description

This function implements a Monte Carlo (randomisation) test for a basic or robust GW PCA with the bandwidth pre-specified and constant. The test evaluates whether the GW eigenvalues vary significantly across space for the first component only.

Usage

```
montecarlo.gwpca.1(data, bw, vars, k = 2, nsims=99,robust = FALSE, kernel = "bisquare",
                   adaptive = FALSE,  p = 2, theta = 0, longlat = F, dMat)
```

Arguments

- `data` a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
- `bw` bandwidth used in the weighting function, possibly calculated by bw.gwpca;fixed (distance) or adaptive bandwidth(number of nearest neighbours)
- `vars` a vector of variable names to be evaluated
- `k` the number of retained components; k must be less than the number of variables
- `nsims` the number of simulations for MontCarlo test
- `robust` if TRUE, robust GWPCA will be applied; otherwise basic GWPCA will be applied
- `kernel` function chosen as follows:

  gaussian: wgt = exp(-.5*(vdist/bw)^2);

  exponential: wgt = exp(-vdist/bw);

  bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;

  tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;

  boxcar: wgt=1 if dist < bw, wgt=0 otherwise
- `adaptive` if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
- `p` the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
- `theta` an angle in radians to rotate the coordinate system, default is 0
- `longlat` if TRUE, great circle distances will be calculated
- `dMat` a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

A list of components:

`actual` the observed standard deviations (SD) of eigenvalues

`sims` a vector of the simulated SDs of eigenvalues

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

Examples

```
## Not run:
data(DubVoter)
DM<-gw.dist(dp.locat=coordinates(Dub.voter))
gmc.res<-montecarlo.gwpca.1(data=Dub.voter, vars=c("DiffAdd", "LARent",
"SC1", "Unempl", "LowEduc"), bw=20,dMat=DM,adaptive=TRUE)
gmc.res
plot(gmc.res)

## End(Not run)
```

---

montecarlo.gwpca.2 *Monte Carlo (randomisation) test for significance of GW PCA eigenvalue variability for the first component only - option 2*

Description

This function implements a Monte Carlo (randomisation) test for a basic or robust GW PCA with the bandwidth automatically re-selected via the cross-validation approach. The test evaluates whether the GW eigenvalues vary significantly across space for the first component only.

Usage

```
montecarlo.gwpca.2(data, vars, k = 2, nsims=99,robust = FALSE, kernel = "bisquare",
                    adaptive = FALSE,  p = 2, theta = 0, longlat = F, dMat)
```

Arguments

`data` a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**

`vars` a vector of variable names to be evaluated

`k` the number of retained components; k must be less than the number of variables

`nsims` the number of simulations for MontCarlo test

`robust` if TRUE, robust GWPCA will be applied; otherwise basic GWPCA will be applied

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function gw.dist

Value

A list of components:

actual	the observed standard deviations (SD) of eigenvalues
sims	a vector of the simulated SDs of eigenvalues

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

Examples

```
## Not run:
data(DubVoter)
DM<-gw.dist(dp.locat=coordinates(Dub.voter))
gmc.res.autow<-montecarlo.gwpca.2(data=Dub.voter, vars=c("DiffAdd", "LARent",
"SC1", "Unempl", "LowEduc"), dMat=DM,adaptive=TRUE)
gmc.res.autow
plot.mcsims(gmc.res.autow)

## End(Not run)
```

---

montecarlo.gwr — *Monte Carlo (randomisation) test for significance of GWR parameter variability*

Description

This function implements a Monte Carlo (randomisation) test to test for significant (spatial) variability of a GWR model's parameters or coefficients.

Usage

```
montecarlo.gwr(formula, data = list(),nsims=99, kernel="bisquare",adaptive=F, bw,
                         p=2, theta=0, longlat=F,dMat)
```

Arguments

`formula`	Regression model formula of a formula object
`data`	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
`nsims`	the number of randomisations

`kernel`	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

`adaptive`	if TRUE calculate an adaptive kernel where the bandwidth (bw) corresponds to the number of nearest neighbours (i.e. adaptive distance); default is FALSE, where a fixed kernel is found (bandwidth is a fixed distance)
`bw`	bandwidth used in the weighting function, possibly calculated by `bw.gwr`
`p`	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
`theta`	an angle in radians to rotate the coordinate system, default is 0
`longlat`	if TRUE, great circle distances will be calculated
`dMat`	a pre-specified distance matrix, it can be calculated by the function `gw.dist`

Value

`pmat`	A vector containing p-values for all the GWR parameters

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Brunsdon C, Fotheringham AS, Charlton ME (1998) Geographically weighted regression - modelling spatial non-stationarity. Journal of the Royal Statistical Society, Series D-The Statistician 47(3):431-443

Charlton, M, Fotheringham, S, and Brunsdon, C (2007), GWR3.0.

Examples

```
## Not run:
data(LondonHP)
DM<-gw.dist(dp.locat=coordinates(londonhp))
bw<-bw.gwr(PURCHASE~FLOORSZ,data=londonhp,dMat=DM, kernel="gaussian")
#See any difference in the next two commands and why?
res.mont1<-montecarlo.gwr(PURCHASE~PROF+FLOORSZ, data = londonhp,dMat=DM,
nsim=99, kernel="gaussian", adaptive=FALSE, bw=3000)
res.mont2<-montecarlo.gwr(PURCHASE~PROF+FLOORSZ, data = londonhp,dMat=DM,
nsim=99, kernel="gaussian", adaptive=FALSE, bw=300000000000)

## End(Not run)
```

---

montecarlo.gwss — *Monte Carlo (randomisation) test for gwss*

Description

This function implements Monte Carlo (randomisation) tests for the GW summary statistics found in gwss.

Usage

```
montecarlo.gwss(data, vars, kernel = "bisquare",
                adaptive = FALSE, bw, p = 2, theta = 0, longlat = F,
                dMat, quantile=FALSE,nsim=99)
```

Arguments

data	a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp**
vars	a vector of variable names to be summarized
bw	bandwidth used in the weighting function

kernel	function chosen as follows:
	gaussian: wgt = exp(-.5*(vdist/bw)^2);
	exponential: wgt = exp(-vdist/bw);
	bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;
	tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;
	boxcar: wgt=1 if dist < bw, wgt=0 otherwise

adaptive	if TRUE calulate the adaptive kernel, and bw correspond to the number of nearest neighbours, default is FALSE.
p	the power of the Minkowski distance, default is 2, i.e. the Euclidean distance
theta	an angle in radians to rotate the coordinate system, default is 0
longlat	if TRUE, great circle distances will be calculated
dMat	a pre-specified distance matrix, it can be calculated by the function `gw.dist`
quantile	if TRUE, median, interquartile range, quantile imbalance will be calculated
nsim	default 99, the number of randomisations

Value

test — probability of the test statistics of the GW summary statistics; if p<0.025 or if p>0.975 then the true local summary statistics can be said to be significantly different (at the 0.95 level) to such a local summary statistics found by chance.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Fotheringham S, Brunsdon, C, and Charlton, M (2002), Geographically Weighted Regression: The Analysis of Spatially Varying Relationships, Chichester: Wiley.

Brunsdon C, Fotheringham AS, Charlton ME (2002) Geographically weighted summary statistics - a framework for localised exploratory data analysis. Computers, Environment and Urban Systems 26:501-524

Harris P, Brunsdon C (2010) Exploring spatial variation and spatial relationships in a freshwater acidification critical load data set for Great Britain using geographically weighted summary statistics. Computers & Geosciences 36:54-70

Examples

```
## Not run:
data(LondonHP)
DM<-gw.dist(dp.locat=coordinates(londonhp))
test.lss<-montecarlo.gwss(data=londonhp, vars=c("PURCHASE","FLOORSZ"), bw=5000,
          kernel ="gaussian", dMat=DM,nsim=99)
test.lss

## End(Not run)
```

---

plot.mcsims — *Plot the results from the Monte Carlo (randomisation) test of GWPCA*

Description

This function plots the results from the functions montecarlo.gwpca.1 and montecarlo.gwpca.2.

Usage

```
## S3 method for class 'mcsims'
plot(x, sname="SD of local eigenvalues from randomisations", ...)
```

Arguments

`x`	an object of class "mcsims", returned by the function montecarlo.gwpca.1 or montecarlo.gwpca.2
`sname`	the label for the observed value on the plot
`...`	arguments passed through (unused)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

`print.ggwrm` *Print the output of the function* `gwr.generalised`

Description

This function prints out the generalised GWR results from the function `gwr.generalised` with reference to the layout of the GWR3.0 software.

Usage

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

Arguments

`x`	an object of class "ggwrm", returned by the function gwr.generalised
`...`	arguments passed through (unused)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Charlton, M, Fotheringham, S, and Brunsdon, C (2007), GWR3.0.

print.gwrlcr   *Print the output of the function* gwr.lcr

Description

This function prints out the results from the function gwr.lcr.

Usage

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

Arguments

x	an object of class "gwrlcr", returned by the function gwr.lcr
...	arguments passed through (unused)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

print.gwrm   *Print the output of the function* gwr.basic

Description

This function prints out the GWR results from the function gwr.basic with reference to the layout of the GWR3.0 software.

Usage

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

Arguments

x	an object of class "gwrm", returned by the function gwr.basic
...	arguments passed through (unused)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

References

Charlton, M, Fotheringham, S, and Brunsdon, C (2007), GWR3.0.

print.gwrm.pred — *Print the output of the function* gwr.predict

Description

This function prints out the GWR results from the function gwr.predict

Usage

```
## S3 method for class 'gwrm.pred'
print(x, ...)
```

Arguments

x	an object of class “gwrm.pred”, returned by the function gwr.predict
...	arguments passed through (unused)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

print.gwss — *Print the output of the function* gwss

Description

This function prints out the results from gwss.

Usage

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

Arguments

x	an object of class “gwss”, returned by the function gwss
...	arguments passed through (unused)

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

`USelect` *Results of the 2004 US presidential election at the county level(SpatialPolygonsDataFrame)*

Description

Results of the 2004 US presidential election at the county level, together with five socio-economic (census) variables.

Usage

```
data(USelect)
```

Format

A SpatialPolygonsDataFrame with 3111 electoral divisions on the following 6 variables.

winner Categorical variable with three classes: i) Bush, ii) Kerry and iii) Borderline (supporting ratio for a candidate ranges from 0.45 to 0.55)

unemployed percentage unemployed

pctcoled percentage of adults over 25 with 4 or more years of college education

PEROVER65 percentage of persons over the age of 65

pcturban percentage urban

WHITE percentage white

References

ROBINSON, A. C. (2013). Geovisualization of the 2004 Presidential Election. In: NATIONAL INSTITUTES OF HEALTH, P. S. U. (ed.). Penn State: http://www.personal.psu.edu/users/a/c/acr181/election.html.

FOLEY, P. & DEMSAR, U. (2012). Using geovisual analytics to compare the performance of geographically weighted discriminant analysis versus its global counterpart, linear discriminant analysis. International Journal of Geographical Information Science, 27, 633-661.

Examples

```
data(USelect)
ls()
```

writeGWR — *Write the GWR results*

Description

This function writes the calibration result of function gwr.basic to a text file and shape files

Usage

```
writeGWR(x,fn="GWRresults")
```

Arguments

`x`	an object of class "gwrm", returned by the function gwr.basic
`fn`	file name for the written results, by default the output files can be found in the working directory, "GWRresults.txt", "GWRresults(.shp, .shx, .dbf)"

Note

The projection file is missing for the writen shapefiles.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

See Also

`writeGWR.shp`

writeGWR.shp — *Write GWR results as shape files*

Description

This function writes a spatial data frame of the calibration result of function gwr.basic as shape files

Usage

```
writeGWR.shp(x,fn="GWRresults")
```

Arguments

`x`	an object of class "gwrm", returned by the function gwr.basic
`fn`	file name for the written results, by default the output files can be found in the working directory, "GWRresults(.shp, .shx, .dbf)"

Note

The projection file is missing for the written shapefiles.

Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

See Also

writeGWR

Index