# Package 'GWmodel'

April 12, 2021

**Type** Package

**Version** 2.2-5

**Date** 2021-04-11

**Title** Geographically-Weighted Models

**Depends** R (>= 3.0.0),maptools (>= 0.5-2), robustbase,sp (> 1.4-0),Rcpp,spatialreg

**Imports** methods, grDevices, stats,graphics,spacetime,spdep,FNN

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** mvoutlier, RColorBrewer, gstat,spData

**Description** 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 (Brunsdon et al., 2002)<doi: 10.1016/s0198-9715(01)00009-6>, GW principal components analysis (Harris et al., 2011)<doi: 10.1080/13658816.2011.554838>, GW discriminant analysis (Brunsdon et al., 2007)<doi: 10.1111/j.1538-4632.2007.00709.x> and various forms of GW regression (Brunsdon et al., 1996)<doi: 10.1111/j.1538-4632.1996.tb00936.x>; 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], Daisuke Murakami[aut],Isabella Gollini[ctb], Yigong Hu[ctb], Fiona H Evans[ctb]

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

**License** GPL (>= 2)

**Repository** CRAN

**URL** http://gwr.nuim.ie/

**NeedsCompilation** yes

**SystemRequirements** GNU make

**Date/Publication** 2021-04-12 14:40:06 UTC

# 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. In particular, the high-performence computing technologies, including multi-thread and CUDA techniques are started to be adopted for efficient calibrations.

## Details

| | |
|---|---|
| Package: | GWmodel |
| Type: | Package |
| Version: | 2.2-5 |
| Date: | 2021-04-11 |
| License: | GPL (>=2) |
| LazyLoad: | yes |

## Note

Acknowledgements: We gratefully acknowledge support from National Natural Science Foundation of China (42071368); Science Foundation Ireland under the National Development Plan through the award of a Strategic Research Centre grant 07-SRC-I1168.

Beta versions can always be found at `https://github.com/lbb220/GWmodel`, which includes all the newly developed functions for GW models.

For latest tutorials on using GWmodel please go to: `https://rpubs.com/gwmodel`

### Author(s)

Binbin Lu, Paul Harris, Martin Charlton, Chris Brunsdon, Tomoki Nakaya, Daisuke Murakami,Isabella Gollini[ctb], Yigong Hu[ctb], Fiona H Evans[ctb]

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, `https://www.jstatsoft.org/article/view/v063i17`

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, `https://www.tandfonline.com/doi/abs/10.1080/10095020.2014.917453`

---

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

---

### Description

A function for automatic 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |
| `adaptive` | if TRUE calculate an adaptive kernel where the bandwidth 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 all GW models

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

## bw.gtwr — *Bandwidth selection for GTWR*

---

### Description

A function for automatic bandwidth selection to calibrate a GTWR model

### Usage

```
bw.gtwr(formula, data, obs.tv, approach="CV",kernel="bisquare",adaptive=FALSE,
        p=2, theta=0, longlat=F,lamda=0.05,t.units = "auto",ksi=0, st.dMat,
        verbose=T)
```

### Arguments

| | |
|---|---|
| formula | Regression model formula of a formula object |
| data | a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp** |
| obs.tv | a vector of time tags for each observation, which could be numeric or of POSIXlt class |
| approach | specified by CV for cross-validation approach or by AIC corrected (AICc) approach |

| | |
|---|---|
| kernel | function chosen as follows:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |
| lamda | an parameter between 0 and 1 for calculating spatio-temporal distance |
| t.units | character string to define time unit |
| ksi | an parameter between 0 and PI for calculating spatio-temporal distance, see details in Wu et al. (2014) |
| st.dMat | a pre-specified spatio-temporal distance matrix |
| verbose | logical variable to define whether show the selection procedure |

## Value

Returns the adaptive or fixed distance bandwidth

## Note

The function is developed according to the articles by Huang et al. (2010) and Wu et al. (2014).

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

## References

Huang, B., Wu, B., & Barry, M. (2010). Geographically and temporally weighted regression for modeling spatio-temporal variation in house prices. International Journal of Geographical Information Science, 24, 383-401.

Wu, B., Li, R., & Huang, B. (2014). A geographically and temporally weighted autoregressive model with application to housing prices. International Journal of Geographical Information Science, 28, 1186-1204.

Fotheringham, A. S., Crespo, R., & Yao, J. (2015). Geographical and Temporal Weighted Regression (GTWR). Geographical Analysis, 47, 431-452.

bw.gwda *Bandwidth selection for GW Discriminant Analysis*

## Description

A function for automatic bandwidth selection for GW Discriminant Analysis using a cross-validation approach only

## 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 | 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, a weighted quadratic discriminant analysis will be applied; otherwise a weighted linear discriminant analysis will be applied |
| kernel | function chosen as follows:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 all GW models

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

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

---

## Description

A function for automatic bandwidth selection to calibrate a basic or robust GWPCA via a cross-validation approach only

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |

| | |
|---|---|
| adaptive | if TRUE calculate an adaptive kernel where the bandwidth 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 all GW models

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

## References

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 47: 146-172

---

bw.gwr *Bandwidth selection for basic GWR*

---

## Description

A function for automatic 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,
       parallel.method=F,parallel.arg=NULL)
```

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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` |
| parallel.method | FALSE as default, and the calibration will be conducted traditionally via the serial technique, "omp": multi-thread technique with the OpenMP API, "cluster": multi-process technique with the **parallel** package, "cuda": parallel computing technique with CUDA |
| parallel.arg | if parallel.method is not FALSE, then set the argument by following: if parallel.method is "omp", parallel.arg refers to the number of threads used, and its default value is the number of cores - 1; if parallel.method is "cluster", parallel.arg refers to the number of R sessions used, and its default value is the number of cores - 1; if parallel.method is "cuda", parallel.arg refers to the number of calibrations included in each group, but note a too large value may cause the overflow of GPU memory. |

## 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 all GW models

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

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

---

### Description

A function for automatic bandwidth selection for gwr.lcr via a cross-validation approach only

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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. 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 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

### 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 all GW models

### Author(s)

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

---

## bw.gwss.average *Bandwidth selection for GW summary averages*

---

### Description

A function for automatic bandwidth selections to calculate GW summary averages, including means and medians, via a cross-validation approach.

### Usage

```
bw.gwss.average(data, summary.locat, vars, 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** |
| 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 |

| | |
|---|---|
| kernel | function chosen as follows:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 bandwidths (in a two-column matrix) for calculating the averages of each variable.

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

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 for England and Wales from 2001 with 9 hedonic (explanatory) variables.

## 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 the England and Wales house price data EWHP.

### Usage

```
data(EWOutline)
```

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

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

### Description

Census data from the county of Georgia, USA

### 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 census data with boundaries for mapping

## Usage

```
data(GeorgiaCounties)
```

## Details

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

## 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.basic *Generalised GWR models with Poisson and Binomial options*

## Description

This function implements generalised GWR

## Usage

```
ggwr.basic(formula, data, regression.points, bw, family =
                 "poisson", kernel = "bisquare", adaptive = FALSE, cv =
                 T, tol = 1e-05, maxiter = 20, p = 2, theta = 0,
                 longlat = F, dMat, dMat1)

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

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |
| adaptive | if TRUE calculate an adaptive kernel where the bandwidth 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) |
| 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 |
| 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 between regression points and observations, it can be calculated by the function `gw.dist` |
| `dMat1` | a square distance matrix between each pair of observations, it can be calculated by the function `gw.dist` |
| `x` | an object of class "ggwrm", returned by the function gwr.generalised |
| `...` | arguments passed through (unused) |

## 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 |

## Note

Note that this function calibrates a Generalised GWR model via an approximating algorithm, which is different from the back-fitting algorithm used in the GWR4 software by Tomoki Nakaya.

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

## References

Nakaya, T., A. S. Fotheringham, C. Brunsdon & M. Charlton (2005) Geographically weighted Poisson regression for disease association mapping. Statistics in Medicine, 24, 2695-2717.

Nakaya, T., M. Charlton, S. Fotheringham & C. Brunsdon. 2009. How to use SGWRWIN (GWR4.0). Maynooth, Ireland: National Centre for Geocomputation.

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<-ggwr.basic(BATH2~FLOORSZ, bw=bw.f1,data=londonhp, dMat=DM)
bw.f2 <- bw.ggwr(BATH2~FLOORSZ,data=londonhp, dMat=DM,family ="binomial")
res.binomial<-ggwr.basic(BATH2~FLOORSZ, bw=bw.f2,data=londonhp, dMat=DM,
              family ="binomial")

## End(Not run)
```

---

## 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. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 individual cross-validation score at each observation location, for a generalised GWR model, for a specified bandwidth. These data can be mapped to detect unusually high or low cross-validations scores.

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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>

---

gtwr *Geographically and Temporally Weighted Regression*

---

## Description

A function for calibrating a Geographically and Temporally Weighted Regression (GTWR) model.

## Usage

```
gtwr(formula, data, regression.points, obs.tv, reg.tv, st.bw, kernel="bisquare",
     adaptive=FALSE, p=2, theta=0, longlat=F,lamda=0.05,t.units = "auto",ksi=0,
     st.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**; Note that no diagnostic information will returned if it is assigned |
| obs.tv | a vector of time tags for each observation, which could be numeric or of POSIXlt class |
| reg.tv | a vector of time tags for each regression location, which could be numeric or of POSIXlt class |

| | |
|---|---|
| `st.bw` | spatio-temporal 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |
| `lamda` | an parameter between 0 and 1 for calculating spatio-temporal distance |
| `t.units` | character string to define time unit |
| `ksi` | an parameter between 0 and PI for calculating spatio-temporal distance, see details in Wu et al. (2014) |
| `st.dMat` | a pre-specified spatio-temporal distance matrix |

## Value

A list of class “gtwrm”:

| | |
|---|---|
| `GTW.arguments` | a list class object including the model fitting parameters for generating the report file |
| `GTW.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, GTWR 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. |

## Note

The function implements GTWR model proposed by Huang et al. (2010) and Wu et al. (2014).

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

### References

Huang, B., Wu, B., & Barry, M. (2010). Geographically and temporally weighted regression for modeling spatio-temporal variation in house prices. International Journal of Geographical Information Science, 24, 383-401.

Wu, B., Li, R., & Huang, B. (2014). A geographically and temporally weighted autoregressive model with application to housing prices. International Journal of Geographical Information Science, 28, 1186-1204.

Fotheringham, A. S., Crespo, R., & Yao, J. (2015). Geographical and Temporal Weighted Regression (GTWR). Geographical Analysis, 47, 431-452.

---

## `gw.dist` *Distance matrix calculation*

---

### Description

Calculate a distance vector(matrix) between any GW model calibration point(s) 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 locally 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.

## 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

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 47: 146-172

---

## `gw.weight` *Weight matrix calculation*

---

### Description

Calculate a weight vector(matrix) from a distance vector(matrix).

### Usage

```
gw.weight(vdist,bw,kernel,adaptive=FALSE)
```

### Arguments

- `vdist` a distance matrix or vector
- `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)

### Value

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

### 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 the 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. The function will be the same as a global application function (i.e. all weights are 1) when the adaptive bandwidth is equal to or larger than the number of observations when using the boxcar kernel function.

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

## gwda — *GW Discriminant Analysis*

---

### Description

This function implements GW discriminant analysis, where location-wise probabilities and their associated entropy are also calculated.

### 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` | 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |
| `adaptive` | if TRUE calculate an adaptive kernel where the bandwidth 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

An object of class "gwda". This includes a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object, SDF, (see package "sp") with, following the use of new version of gwda, the probabilities for each level, the highest probabiliity and the entropy of the probabilities in its "data" slot.

### 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

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

### Examples

```
## Not run:
 data(USelect)
 dMat <- gw.dist(coordinates(USelect2004))
 bw <- bw.gwda(winner~unemploy+pctcoled+PEROVER65+pcturban+WHITE,data=USelect2004,
 adaptive=TRUE,dMat=dMat)
 ge.gwda <- gwda(winner~unemploy+pctcoled+PEROVER65+pcturban+WHITE,data=USelect2004,
 bw=bw,adaptive=TRUE,dMat=dMat)
 table(USelect2004$winner,ge.gwda$SDF$group.predicted)
 spplot(ge.gwda$SDF, "entropy")

## End(Not run)
```

---

## 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, scores=F,
                    dMat)
## S3 method for class 'gwpca'
print(x, ...)
```

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |
| `adaptive` | if TRUE calculate an adaptive kernel where the bandwidth 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. |
| `scores` | if scores = TRUE, the scores of the supplied data on the principal components will be calculated. |
| `dMat` | a pre-specified distance matrix, it can be calculated by the function `gw.dist` |
| `x` | an object of class “gwpca”, returned by the function `gwpca` |
| `...` | arguments passed through (unused) |

### Value

A list of class “gwpca”:

| | |
|---|---|
| `GW.arguments` | a list class object including the model fitting parameters for generating the report file |
| `pca` | an object of class inheriting from “princomp”, see princomp. |
| `loadings` | the localised loadings |

| | |
|---|---|
| SDF | a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with local proportions of variance for each principle components, cumulative proportion and winning variable for the 1st principle component in its "data" slot. |
| gwpca.scores | the localised scores of the supplied data on the principal components |
| var | The local amount of variance accounted for by each component |
| CV | Vector of cross-validation data |
| timings | starting and ending time. |

## 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 47: 146-172

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

### Description

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

### Usage

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

### Arguments

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

### Note

The function “check.components” (in the early versions of GWmodel) has been renamed as “gwpca.check.components”, while the old name is still kept valid.

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

### See Also

gwpca.glyph.plot

## 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. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 individual cross-validation score at each observation location, for a GWPCA model, for a specified bandwidth. These data can be mapped to detect unusually high or low cross-validations scores.

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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>

---

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

---

### Description

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

### Usage

```
gwpca.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 |

### Note

The function "glyph.plot" (in the early versions of GWmodel) has been renamed as "gwpca.glyph.plot", while the old name is still kept valid.

### References

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

---

## `gwpca.montecarlo.1` *Monte Carlo (randomisation) test for significance of GWPCA 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

```
gwpca.montecarlo.1(data, bw, vars, k = 2, nsims=99,robust = FALSE, kernel = "bisquare",
                   adaptive = FALSE,  p = 2, theta = 0, longlat = F, dMat)
## S3 method for class 'mcsims'
plot(x, sname="SD of local eigenvalues from randomisations", ...)
```

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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` |
| `x` | an object of class "mcsims", returned by the function gwpca.montecarlo.1 or gwpca.montecarlo.2 |
| `sname` | the label for the observed value on the plot |
| `...` | arguments passed through (unused) |

## Value

A list of components:

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

## Note

The function "montecarlo.gwpca.1" (in the early versions of GWmodel) has been renamed as "gwpca.montecarlo.1", while the old name is still kept valid.

## 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

### Examples

```
## Not run:
data(DubVoter)
DM<-gw.dist(dp.locat=coordinates(Dub.voter))
gmc.res<-gwpca.montecarlo.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)
```

---

## gwpca.montecarlo.2 *Monte Carlo (randomisation) test for significance of GWPCA 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

```
gwpca.montecarlo.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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |

## Note

The function “montecarlo.gwpca.2” (in the early versions of GWmodel) has been renamed as “gwpca.montecarlo.2”, while the old name is still kept valid.

## 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

## Examples

```
## Not run:
data(DubVoter)
DM<-gw.dist(dp.locat=coordinates(Dub.voter))
gmc.res.autow<-gwpca.montecarlo.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)
```

---

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=F, W.vect=NULL,
parallel.method=FALSE,parallel.arg=NULL)
## S3 method for class 'gwrm'
print(x, ...)
```

### 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**; Note that no diagnostic information will returned if it is assigned

`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

`x` an object of class "gwrm", returned by the function `gwr.basic`

`parallel.method` FALSE as default, and the calibration will be conducted traditionally via the serial technique, "omp": multi-thread technique with the OpenMP API, "cluster": multi-process technique with the **parallel** package, "cuda": parallel computing technique with CUDA `parallel.arg` — if parallel.method is not FALSE, then set the argument by following: if parallel.method is "omp", parallel.arg refers to the number of threads used, and its default value is the number of cores - 1; if parallel.method is "cluster", parallel.arg refers to the number of R sessions used, and its default value is the number of cores - 1; if parallel.method is "cuda", parallel.arg refers to the number of calibrations included in each group, but note a too large value may cause the overflow of GPU memory.

`...` — arguments passed through (unused)

## 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

Requirements of using CUDA for high-performence computation in GWR functions:

To run GWR-CUDA (i.e. parallel.method is pecified as “cuda”) with `gwr.basic` , `bw.gwr` and `gwr.model.selection`, the following conditions are required:

1. There is at least one NVIDIA GPU supporting CUDA equipped on user’s computer.

2. CUDA (>10.2) are installed with the environment variable ‘CUDA_HOME‘ set properly.

3. The package should re-built from source. - For Linux user, ‘GWmodelCUDA‘ will be automatically built if CUDA toolkit could be detected by the complier. - For Windows user, ‘GWmodelCUDA.dll‘ and ‘GWmodelCUDA.lib‘ will be automatically downloaded; however, we would recommend users to build the ‘GWmodelCUDA‘ library manually to avoid some potentially unknown issues, and an ‘CMakeLists.txt‘ file is provided for this procedure.

If any condition above is not satisfied, the GWR-CUDA will not work even though the “parallel.method” is specified as “cuda”.

## 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

OpenMP: https://www.openmp.org/

CUDA: https://developer.nvidia.com/cuda-zone

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.


## 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))

## 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
## End(Not run)
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)
## Not run:
  ############HP-GWR test code
  simulate.data.generator <- function(data.length) {
  x1 <- rnorm(data.length)
  x2 <- rnorm(data.length)
  x3 <- rnorm(data.length)
  lon <- rnorm(data.length, mean = 533200, sd = 10000)
  lat <- rnorm(data.length, mean = 159400, sd = 10000)
  y <- x1 + 5 * x2 + 2.5 * x3 + rnorm(data.length)
  simulate.data <- data.frame(y = y, x1 = x1, x2 = x2, x3 = x3, lon = lon, lat = lat)
  coordinates(simulate.data) <- ~ lon + lat
  names(simulate.data)
  return(simulate.data)
}
simulate.data <- simulate.data.generator(10000)
adaptive = TRUE

## GWR (not parallelized)
bw.CV.s <- bw.gwr(data = simulate.data, formula = y ~ x1 + x2 + x3, approach="CV",
                  kernel = "gaussian", adaptive = adaptive, parallel.method = FALSE)
model.s <- gwr.model.selection(DeVar = "y", InDeVars = c("x1", "x2", "x3"), data = simulate.data,
                         bw = bw.CV.s, approach="AIC", kernel = "gaussian", adaptive = T,
                              parallel.method = FALSE)
system.time(
  betas.s <- gwr.basic(data = simulate.data, formula = y ~ x1 + x2 + x3, bw = bw.CV.s,
                       kernel = "gaussian", adaptive = TRUE)
)

## GWR-Omp
bw.CV.omp <- bw.gwr(data = simulate.data, formula = y ~ x1 + x2 + x3, approach="CV",
                    kernel = "gaussian", adaptive = adaptive, parallel.method = "omp")
model.omp <- gwr.model.selection(DeVar = "y", InDeVars = c("x1", "x2", "x3"), data = simulate.data,
                        bw = bw.CV.omp, approach="AIC", kernel = "gaussian", adaptive = T,
                                parallel.method = "omp")
system.time(
  betas.omp <- gwr.basic(data = simulate.data, formula = y ~ x1 + x2 + x3, bw = bw.CV.omp,
                        kernel = "gaussian", adaptive = T, parallel.method = "omp"))
```

```
## GWR-CUDA
bw.CV.cuda <- bw.gwr(data = simulate.data, formula = y ~ x1 + x2 + x3, approach="CV",
                     kernel = "gaussian", adaptive = adaptive, parallel.method = "cuda",
                     parallel.arg = 6*16)
model.cuda <- gwr.model.selection(DeVar = "y", InDeVars = c("x1", "x2", "x3"), data = simulate.data,
                       bw = bw.CV.cuda, approach="AIC", kernel = "gaussian", adaptive = T,
                                parallel.method = "cuda", parallel.arg = 6*16)
system.time(
  betas.cuda <- gwr.basic(data = simulate.data, formula = y ~ x1 + x2 + x3, bw = bw.CV.cuda,
                          kernel = "gaussian", adaptive = T, parallel.method = "cuda",
                          parallel.arg = 6*8))

## End(Not run)
```

---

gwr.bootstrap *Bootstrap GWR*

---

## Description

This function implements bootstrap methods to test for coefficient variability found from GWR under model assumptions for each of four null hypotheses: MLR, ERR, SMA and LAG models. Global test statistic results are found, as well local observation-specific test results that can be mapped.

## Usage

```
gwr.bootstrap(formula, data, kernel = "bisquare", approach = "AIC",
                 R = 99, k.nearneigh = 4, adaptive = FALSE, p = 2,
                 theta = 0, longlat = FALSE, dMat, verbose = FALSE,
                 parallel.method = FALSE, parallel.arg = NULL)

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

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |

| | |
|---|---|
| `approach` | specified by CV for cross-validation approach or by AIC corrected (AICc) approach |
| `R` | number of random samples reapted in the bootstrap procedure |
| `k.nearneigh` | number of nearest neighbours concerned in calbrating ERR, SMA and LAG models |
| `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` |
| `verbose` | if TRUE and bandwidth selection is undertaken, the bandwidth searches are reported |
| `x` | an object of class “gwrbsm”, returned by the function gwr.bootstrap |
| `parallel.method` | FALSE as default, and the calibration will be conducted traditionally via the serial technique, "omp": multi-thread technique with the OpenMP API, "cluster": multi-process technique with the **parallel** package, "cuda": parallel computing technique with CUDA |
| `parallel.arg` | if parallel.method is not FALSE, then set the argument by following: if parallel.method is "omp", parallel.arg refers to the number of threads used, and its default value is the number of cores - 1; if parallel.method is "cluster", parallel.arg refers to the number of R sessions used, and its default value is the number of cores - 1; if parallel.method is "cuda", parallel.arg refers to the number of calibrations included in each group, but note a too large value may cause the overflow of GPU memory. |
| `...` | arguments passed through (unused) |

## Value

A list of class “gwrbsm”:

| | |
|---|---|
| `formula` | Regression model formula of a formula object |
| `results` | modified statistics reported from comparisons between GWR and MLR, ERR, SMA and LAG |
| `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 bootstrap p-values in its “data” slot. |
| `timings` | starting and ending time. |
| `this.call` | the function call used. |

## Note

This function implements the bootstrap methods introduced in Harris et al. (2017). It provides a global test statistic (the modified one given in Harris et al. 2017) and a complementary localised version that can be mapped. The bootstrap methods test for coefficient variability found from GWR under model assumptions for each of four null hypotheses: i) multiple linear regression model (MLR); ii) simultaneous autoregressive error model (ERR); iii) moving average error model (SMA) and iv) simultaneous autoregressive lag model (LAG).

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

## References

Harris, P., Brunsdon, C., Lu, B., Nakaya, T., & Charlton, M. (2017). Introducing bootstrap methods to investigate coefficient non-stationarity in spatial regression models. Spatial Statistics, 21, 241-261.

## Examples

```
## Not run:
#Example with the Georgia educational attainment data
data(Georgia)
data(GeorgiaCounties)
coords <- cbind(Gedu.df$X, Gedu.df$Y)
Gedu.spdf <- SpatialPointsDataFrame(coords, Gedu.df)
#Make a SpatialPolygonDataFrame
require(RColorBrewer)
gSRDF <- SpatialPolygonsDataFrame(polygons(Gedu.counties), over(Gedu.counties,
                                  Gedu.spdf),match.ID=T)
mypalette.1 <- brewer.pal(11,"Spectral")
X11(width=9,height=8)
spplot(gSRDF, names(gSRDF)[c(5,7:9)], col.regions=mypalette.1,
cuts=10, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Georgia educational attainment predictor data")))
bsm.res <- gwr.bootstrap(PctBach~PctRural+PctEld+PctFB+PctPov, gSRDF,
                         R=999, longlat=T)
bsm.res
#local bootstrap tests with respect to: MLR, ERR, SMA and LAG models.
mypalette.local.test <- brewer.pal(10,"Spectral")
X11(width=12,height=16)
spplot(bsm.res$SDF, names(bsm.res$SDF)[14:17], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the MLR model
                       null hypothesis")))

X11(width=12,height=16)
spplot(bsm.res$SDF, names(bsm.res$SDF)[19:22], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the ERR model
                       null hypothesis")))
```

```
X11(width=12,height=16)
spplot(bsm.res$SDF, names(bsm.res$SDF)[24:27], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the SMA model null
                       hypothesis")))

X11(width=12,height=16)
spplot(bsm.res$SDF, names(bsm.res$SDF)[29:32], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the LAG model null
                       hypothesis")))
################################################################################
#Example with Dublin voter data
data(DubVoter)
X11(width=9,height=8)
spplot(Dub.voter, names(Dub.voter)[c(5,7,9,10)], col.regions=mypalette.1,
cuts=10, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Dublin voter turnout predictor data")))
bsm.res1 <- gwr.bootstrap(GenEl2004~LARent+Unempl+Age18_24+Age25_44, Dub.voter
                         , R=999)
bsm.res1

#local bootstrap tests with respect to: MLR, ERR, SMA and LAG models.
X11(width=11,height=8)
spplot(bsm.res1$SDF, names(bsm.res1$SDF)[14:17], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the MLR model null
                        hypothesis")))
X11(width=11,height=8)
spplot(bsm.res1$SDF, names(bsm.res1$SDF)[19:22], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the ERR model null
                        hypothesis")))
X11(width=11,height=8)
spplot(bsm.res1$SDF, names(bsm.res1$SDF)[24:27], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the SMA model
                            null hypothesis")))
X11(width=11,height=8)
spplot(bsm.res1$SDF, names(bsm.res1$SDF)[29:32], col.regions=mypalette.local.test,
cuts=9, par.settings=list(fontsize=list(text=15)),
main=expression(paste("Local p-values for each coefficient of the LAG model
                            null hypothesis")))

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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

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

---

`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. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.

## Usage

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

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |
| `parallel.method` | Specified by 'FALSE' for serial approach, by '"omp"' for multi-thread approach implemented via OpenMP, by '"cluster"' for multi-process approach implemented via 'parallel' package, by '"cuda"' for parallel approach implemented via CUDA |
| `parallel.arg` | Set the argument for parallel approach. If 'parallel.method' is 'FALSE', there is no need to set its value. If 'parallel.method' is '"omp"', its value is used to set how many threads should be created (default by cores of *cores of CPU* - 1). If 'parallel.method' is '"cluster"', its value is used to set how many R session should be created (default by cores of *cores of CPU* - 1). If 'parallel.method' is '"cuda"', its value is used to set how many samples is included in one group during the calibration. This value should not be too big to avoid the overflow of GPU memory. |

## 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 individual cross-validation score at each observation location, for a basic GWR model, for a specified bandwidth. These data can be mapped to detect unusually high or low cross-validations scores.

## Usage

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

## 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`

`parallel.method`
Specified by ‘FALSE‘ for serial approach, by ‘"omp"‘ for multi-thread approach implemented via OpenMP, by ‘"cluster"‘ for multi-process approach implemented via ‘parallel‘ package, by ‘"cuda"‘ for parallel approach implemented via CUDA

`parallel.arg` Set the argument for parallel approach. If ‘parallel.method‘ is ‘FALSE‘, there is no need to set its value. If ‘parallel.method‘ is ‘"omp"‘, its value is used to set how many threads should be created (default by cores of *cores of CPU* - 1). If ‘parallel.method‘ is ‘"cluster"‘, its value is used to set how many R session should be created (default by cores of *cores of CPU* - 1). If ‘parallel.method‘ is ‘"cuda"‘, its value is used to set how many samples is included in one group during the calibration. This value should not be too big to avoid the overflow of GPU memory.

## 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.hetero` *Heteroskedastic GWR*

---

## Description

This function implements a heteroskedastic GWR model

### 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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)
## S3 method for class 'gwrlcr'
print(x, ...)
```

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |
| x | an object of class "gwrlcr", returned by the function gwr.lcr |
| ... | arguments passed through (unused) |

## 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. It can be used to construct the bandwidth function across all possible bandwidths and compared to that found automatically.

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 individual cross-validation score at each observation location, for a GWR-LCR model, for a specified bandwidth. These data can be mapped to detect unusually high or low cross-validations scores.

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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.mink.approach *Minkovski approach for GWR*

---

## Description

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

## Usage

```
gwr.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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |

## Note

The function “mink.approach” (in the early versions of GWmodel) has been renamed as “gwr.mink.approach”, while the old name is still kept valid.

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

## References

Lu, B, Charlton, M, Brunsdon, C & Harris, P(2016). The Minkowski approach for choosing the distance metric in Geographically Weighted Regression. International Journal of Geographical Information Science, 30(2): 351-368.

---

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

---

## Description

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

## Usage

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

### Arguments

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

### Note

The function “mink.matrixview” (in the early versions of GWmodel) has been renamed as “gwr.mink.matrixview”, while the old name is still kept valid.

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

### References

Lu, B, Charlton, M, Brunsdon, C & Harris, P(2016). The Minkowski approach for choosing the distance metric in Geographically Weighted Regression. International Journal of Geographical Information Science, 30(2): 351-368.

---

## `gwr.mink.pval` *Select the values of p for the Minkowski approach for GWR*

---

### Description

These functions implement heuristics to select the values of p from two intervals: (0, 2] in a ’backward’ direction and (2, Inf) in a ’forward’ direction.

### Usage

```
gwr.mink.pval(formula, data, criterion="AIC", bw, bw.sel.approach = "AIC",
                       adaptive=F, kernel="bisquare", left.interval=0.25,
                  right.interval=0.5,drop.tol=3, theta0=0,verbose=F,nlower = 10)
gwr.mink.pval.forward(formula, data, bw, bw.sel.approach = "AIC",
                       adaptive=F, kernel="bisquare", p.max=Inf,p.min=2,
                       interval=0.5,drop.tol=3, theta0=0,verbose=F,nlower = 10)
gwr.mink.pval.backward(formula, data, bw, bw.sel.approach = "AIC",
                       adaptive=F, kernel="bisquare", p.max=2,p.min=0.1,
                       interval=0.5,drop.tol=3, theta0=0,verbose=F,nlower = 10)
## S3 method for class 'pvlas'
plot(x, ...)
```

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>boxcar: wgt=1 if dist < bw, wgt=0 otherwise |
| left.interval | the step-size for searching the left interval (0, 2] in a 'backward' direction |
| right.interval | the step-size for searching the right interval (2, Inf) in a 'forward' direction |
| p.max | the maximum value of p |
| p.min | the minimum value of p |
| interval | the step-size for searching the given interval in a 'backward' or 'forward' direction |
| drop.tol | an AICc difference threshold to define whether the values of p to be dropped or not |
| theta0 | a fixed rotation angle in radians |
| 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 |
| x | an object of class "pvlas", returned by these functions |
| ... | arguments passed through (unused) |

## Value

A list of:

| | |
|---|---|
| p.vals | a vector of tried values of p |
| cretion.vals | a vector of criterion values (AICc or CV) for tried values of p |
| p.dropped | a vector of boolean to label whether a value of p to be dropped or not: TRUE means to be dropped and FALSE means to be used for the Minkowski approach |

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

### References

Lu, B, Charlton, M, Brunsdon, C & Harris, P(2016). The Minkowski approach for choosing the distance metric in Geographically Weighted Regression. International Journal of Geographical Information Science, 30(2): 351-368.

---

## gwr.mixed *Mixed GWR*

---

### Description

This function implements mixed (semiparametric) 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, dMat.rp)
```

### 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 |
| dMat.rp | a distance matrix when an individual set of regression points are adopted |

## 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. |

## Note

For an alternative formulation of mixed GWR, please refer to GWR 4, which provides useful tools for automatic bandwidth selection. This windows-based software also implements generalised mixed GWR.

The mixed GWR in the latest release of GWmodel (2.0-0) has been revised by Dr. Fiona H Evans from Centre for Digital Agriculture, Murdoch and Curtin Universities in terms of its computational efficiency.

## 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

Nakaya T, Fotheringham AS, Brunsdon C, Charlton M (2005) Geographically Weighted Poisson Regression for Disease Association Mapping, Statistics in Medicine 24: 2695-2717

Nakaya T et al. (2011) GWR4.0, http://gwr.nuim.ie/.

---

`gwr.model.selection` *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

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

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |

`parallel.method`
: Specified by ‘FALSE‘ for serial approach, by ‘"omp"‘ for multi-thread approach implemented via OpenMP, by ‘"cluster"‘ for multi-process approach implemented via ‘parallel‘ package, by ‘"cuda"‘ for parallel approach implemented via CUDA

`parallel.arg`
: Set the argument for parallel approach. If ‘parallel.method‘ is ‘FALSE‘, there is no need to set its value. If ‘parallel.method‘ is ‘"omp"‘, its value is used to set how many threads should be created (default by cores of *cores of CPU* - 1). If ‘parallel.method‘ is ‘"cluster"‘, its value is used to set how many R session should be created (default by cores of *cores of CPU* - 1). If ‘parallel.method‘ is ‘"cuda"‘, its value is used to set how many samples is included in one group during the calibration. This value should not be too big to avoid the overflow of GPU memory.

## 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

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

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

### Description

Sort the results from the GWR model selection function gwr.model.selection

### Usage

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

### Arguments

| | |
|---|---|
| Sorting.list | a list returned by function gwr.model.selection |
| 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 individual levels.

The function "model.sort.gwr" (in the early versions of GWmodel) has been renamed as "gwr.model.sort", while the old name is still kept valid.

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

### See Also

gwr.model.selection, gwr.model.view

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

### Description

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

### Usage

```
gwr.model.view(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 gwr.model.selection |

### Note

The function "model.view.gwr" (in the early versions of GWmodel) has been renamed as "gwr.model.view", while the old name is still kept valid.

### Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

### See Also

gwr.model.selection, gwr.model.sort

### Examples

```
## Not run:
data(LondonHP)
DM<-gw.dist(dp.locat=coordinates(londonhp))
DeVar<-"PURCHASE"
InDeVars<-c("FLOORSZ","GARAGE1","BLDPWW1","BLDPOSTW")
model.sel<-gwr.model.selection(DeVar,InDeVars, data=londonhp,
kernel = "gaussian", dMat=DM,bw=5000)
model.list<-model.sel[[1]]
gwr.model.view(DeVar, InDeVars, model.list=model.list)

## End(Not run)
```

---

## gwr.montecarlo *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

```
gwr.montecarlo(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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |

## Note

The function “montecarlo.gwr” (in the early versions of GWmodel) has been renamed as “gwr.montecarlo”, while the old name is still kept valid.

## 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

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

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<-gwr.montecarlo(PURCHASE~PROF+FLOORSZ, data = londonhp,dMat=DM,
nsim=99, kernel="gaussian", adaptive=FALSE, bw=3000)
res.mont2<-gwr.montecarlo(PURCHASE~PROF+FLOORSZ, data = londonhp,dMat=DM,
nsim=99, kernel="gaussian", adaptive=FALSE, bw=300000000000)

## End(Not run)
```

---

`gwr.multiscale` *Multiscale GWR*

---

## Description

This function implements multiscale GWR to detect variations in regression relationships across different spatial scales. This function can not only find a different bandwidth for each relationship but also (and simultaneously) find a different distance metric for each relationship (if required to do so).

## Usage

```
gwr.multiscale(formula, data, kernel = "bisquare", adaptive = FALSE,
                 criterion = "dCVR", max.iterations = 2000, threshold =
                 1e-05, dMats, var.dMat.indx, p.vals, theta.vals,
                longlat = FALSE, bws0, bw.seled, approach = "AIC", bws.thresholds,
                 bws.reOpts = 5, verbose = F,
                 hatmatrix = T, predictor.centered = rep(T,
                 length(bws0) - 1), nlower = 10, parallel.method = F,
                 parallel.arg = NULL)
## S3 method for class 'multiscalegwr'
print(x, ...)
```

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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)
- `criterion` criterion for determining the convergence of the back-fitting procedure, could be "CVR" or "dCVR", which corespond to the changing value of RSS (CVR) and the differential version (dCVR), respectively; and "dCVR" is used as default.
- `max.iterations` maximum number of iterations in the back-fitting procedure
- `threshold` threshold value to terminate the back-fitting iterations
- `dMats` a list of distance matrices used for estimating each specific parameter
- `var.dMat.indx` index corresponds to a specific distance matrix for each exploratory variable, if dMats is provided
- `p.vals` a collection of positive numbers used as the power of the Minkowski distance
- `theta.vals` a collection of values used as angles in radians to rotate the coordinate system
- `longlat` if TRUE, great circle distances will be calculated
- `bws0` a vector of initializing bandwidths for the back-fitting procedure, of which the length should equal to the number of paramters if specified
- `bw.seled` a vector of boolean variables to determine whether the corresponding bandwidth should be re-selected or not: if TRUE, the corresponding bandwiths for the specific parameters are supposed to be given in bws0; otherwise, the bandwidths for the specific parameters will be selected within the back-fitting iterations.
- `approach` specified by CV for cross-validation approach or by AIC corrected (AICc) approach
- `bws.thresholds` threshold values to define whether the bandwidth for a specific parameter has converged or not
- `bws.reOpts` the number times of continually optimizing each parameter-specific bandwidth even though it meets the criterion of convergence, for avoiding sub-optimal choice due to illusion of convergence;
- `verbose` if TRUE and bandwidth selection is undertaken, the bandwidth searches are reported
- `predictor.centered` a logical vector of length equalling to the number of predictors, and note intercept is not included; if the element is TRUE, the corresponding predictor will be centered.
- `hatmatrix` if TRUE the hatmatrix for the whole model will be calculated, and AICc, adjusted-R2 values will be returned accordingly.
- `nlower` the minmum number of nearest neighbours if an adaptive kernel is used
- `parallel.method` FALSE as default, and the calibration will be conducted traditionally via the serial technique, "omp": multi-thread technique with the OpenMP API, "cluster": multi-process technique with the **parallel** package, "cuda": parallel computing technique with CUDA

`parallel.arg` if parallel.method is not FALSE, then set the argument by following: if parallel.method is "omp", parallel.arg refers to the number of threads used, and its default value is the number of cores - 1; if parallel.method is "cluster", parallel.arg refers to the number of R sessions used, and its default value is the number of cores - 1; if parallel.method is "cuda", parallel.arg refers to the number of calibrations included in each group, but note a too large value may cause the overflow of GPU memory.

`x` an object of class "multiscalegwr", returned by the function gwr.multiscale

`...` arguments passed through (unused)

## Value

A list of class "psdmgwr":

`SDF` a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") integrated with data locations,coefficient estimates from the PSDM GWR model,predicted y values,residuals, coefficient standard errors and t-values in its "data" slot.

`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.

`bws.vars` bandwidths used for all the parameters within the back-fitting procedure

`timings` starting and ending time.

`this.call` the function call used.

## Note

This function implements multiscale GWR to detect variations in regression relationships across different spatial scales. This function can not only find a different bandwidth for each relationship, but also (and simultaneously), find a different distance metric for each relationship (i.e. Parameter-Specific Distance Metric GWR, i.e. PSDM GWR). Note that multiscale GWR (MGWR) has also been referred to as flexible bandwidth GWR (FBGWR) and conditional GWR (CGWR) in the literature. All are one and the same model, but where PSDM-GWR additionally provides a different distance metric option for each relationship. An MGWR model is calibrated if no "dMats" and "p.vals" are specified; a mixed GWR model will be calibrated if an infinite bandwidth and another regular bandwidth are used for estimating the global and local parameters (again when no "dMats" and "p.vals" are specified). In other words, the gwr.multiscale function is specified with Euclidean distances in both cases. Note that the results from this function for a mixed GWR model and gwr.mixed might be different, as a back-fitting algorithm is used in gwr.multiscale, while an approximating algorithm is applied in gwr.mixed. The gwr.mixed function performs better in computational efficiency, but poorer in prediction accuracy.

## Author(s)

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

## References


Yang, W. (2014). An Extension of Geographically Weighted Regression with Flexible Bandwidths. St Andrews, St Andrews, UK.

Lu, B., Harris, P., Charlton, M., & Brunsdon, C. (2015). Calibrating a Geographically Weighted Regression Model with Parameter-specific Distance Metrics. Procedia Environmental Sciences, 26, 109-114.

Lu, B., Brunsdon, C., Charlton, M., & Harris, P. (2017). Geographically weighted regression with parameter-specific distance metrics. International Journal of Geographical Information Science, 31, 982-998.

Fotheringham, A. S., Yang, W. & Kang, W. (2017). Multiscale Geographically Weighted Regression (MGWR). Annals of the American Association of Geographers, 107, 1247-1265.

Yu, H., A. S. Fotheringham, Z. Li, T. Oshan, W. Kang & L. J. Wolf. 2019. Inference in multiscale geographically weighted regression. Geographical Analysis(In press).

Leong, Y.Y., & Yue, J.C. (2017). A modification to geographically weighted regression. International Journal of Health Geographics, 16 (1), 11.

Lu, B., Yang, W. Ge, Y. & Harris, P. (2018). Improvements to the calibration of a geographically weighted regression with parameter-specific distance metrics and bandwidths. Forthcoming Computers, Environment and Urban Systems.

Wolf, L.J, Oshan, T.M, Fotheringham, A.S. (2018). Single and multiscale models of process spatial heterogeneity. Geographical Analysis, 50(3): 223-246.

Murakami, D., Lu, B., Harris, P., Brunsdon, C., Charlton, M., Nakaya, T., & Griffith, D. (2019) The importance of scale in spatially varying coefficient modelling. Forthcoming Annals of the Association of American Geographers.


## Examples

```
data(LondonHP)
EUDM <- gw.dist(coordinates(londonhp))
#No bandwidth is selected, and bws0 values are used
## Not run:
###Similar as the basic GWR
res1<-gwr.multiscale(PURCHASE~FLOORSZ+PROF, data=londonhp, criterion="dCVR",kernel="gaussian",
adaptive=T, bws0=c(100, 100, 100),bw.seled=rep(T, 3), dMats=list(EUDM,EUDM,EUDM))
#FBGWR
res2<-gwr.multiscale(PURCHASE~FLOORSZ+PROF, data=londonhp, criterion="dCVR",kernel="gaussian",
adaptive=T, bws0=c(100, 100, 100), dMats=list(EUDM,EUDM,EUDM))
#Mixed GWR
res3<-gwr.multiscale(PURCHASE~FLOORSZ+PROF, data=londonhp, bws0=c(Inf, 100, 100, Inf),
              bw.seled=rep(T, 3),kernel="gaussian", dMats=list(EUDM,EUDM,EUDM))
#PSDM GWR
res4<- gwr.multiscale(PURCHASE~FLOORSZ+PROF, data=londonhp, kernel="gaussian", p.vals=c(1,2,3))

## End(Not run)
```

`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 unobserved locations are used for prediction).

## Usage

```
gwr.predict(formula, data, predictdata, bw, kernel="bisquare",adaptive=FALSE, p=2,
           theta=0, longlat=F,dMat1, dMat2)
## S3 method for class 'gwrm.pred'
print(x, ...)
```

## 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. SpatialPoints-DataFrame 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |
| `x` | an object of class “gwrm.pred”, returned by the function gwr.predict |
| `...` | arguments passed through (unused) |

### 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

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

```
## Not run:
data(LondonHP)
gwr.pred<-gwr.predict(PURCHASE~FLOORSZ, data=londonhp, bw=2000,kernel = "gaussian")
gwr.pred
#########Global OLS regression results and comparison with gstat functions
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 two robust GWR models.

## Usage

```
gwr.robust(formula, data, 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** |
| `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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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.scalable *Scalable GWR*

---

## Description

This function implements Scalable GWR for large dataset

## Usage

```
gwr.scalable(formula, data, bw.adapt=100, kernel = "gaussian", polynomial = 4,
              p = 2, theta = 0, longlat = F, dMat)
## S3 method for class 'scgwrm'
print(x, ...)
```

## Arguments

| | |
|---|---|
| formula | Regression model formula of a formula object |
| data | a Spatial*DataFrame, i.e. SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package **sp** |
| bw.adapt | adaptive bandwidth (i.e. number of nearest neighbours) used for geographically weighting |
| kernel | Kernel function to calculate the spatial weights, but note only two continuous functions available:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw); |
| polynomial | Degree of the polyunomial to approximate the kernel function, and default is 4. |
| 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 "scgwrm", returned by the function `gwr.scalable` |
| `...` | arguments passed through (unused) |

## Value

A list of class "scgwrm":

| | |
|---|---|
| `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. |

## Author(s)

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

## References

Murakami, D., N. Tsutsumida, T. Yoshida, T. Nakaya & B. Lu (2019) Scalable GWR: A linear-time algorithm for large-scale geographically weighted regression with polynomial kernels. arXiv:1905.00266.

## Examples

```
## Not run:
require(spData)
data(boston)
boston <- boston.c
coordinates(boston) <- ~ LON + LAT
res <- gwr.scalable(formula = MEDV ~ CRIM + ZN + INDUS + CHAS + AGE, data = boston, bw.adapt = 100)
res

## 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 basic 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.

## gwr.write — *Write the GWR results into files*

### Description

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

### Usage

```
gwr.write(x,fn="GWRresults")
gwr.write.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.txt", "GWRresults(.shp, .shx, .dbf)" |

## Note

The projection file is missing for the writen shapefiles.

The functions “writeGWR” and “writeGWR.shp” (in the early versions of GWmodel) have been renamed respectively as “gwr.write” and “gwr.write.shp”, while the old names are still kept valid.

## Author(s)

Binbin Lu <binbinlu@whu.edu.cn>

---

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

## 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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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 |
| x | an object of class "gwss", returned by the function gwss |
| ... | arguments passed through (unused) |

## 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

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


## 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)
```

---

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

---

## Description

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

## Usage

```
gwss.montecarlo(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:<br>gaussian: wgt = exp(-.5*(vdist/bw)^2);<br>exponential: wgt = exp(-vdist/bw);<br>bisquare: wgt = (1-(vdist/bw)^2)^2 if vdist < bw, wgt=0 otherwise;<br>tricube: wgt = (1-(vdist/bw)^3)^3 if vdist < bw, wgt=0 otherwise;<br>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. |

## Note

The function “montecarlo.gwss” (in the early versions of GWmodel) has been renamed as “gwss.montecarlo”, while the old name is still kept valid.

## 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<-gwss.montecarlo(data=londonhp, vars=c("PURCHASE","FLOORSZ"), bw=5000,
          kernel ="gaussian", dMat=DM,nsim=99)
test.lss

## 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.

### 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)
```

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. This data can be used with GW Discriminant Analysis.

## 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)

**unemploy** 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()
```

# Index