Package ‘spgwr’

February 11, 2020

Version 0.6-33
Date 2020-02-10
Title Geographically Weighted Regression
Depends R (>= 2.14), sp (>= 0.8-3), spData (>= 0.2.6.2)
Imports stats, methods
Suggests spdep, parallel, maptools (>= 0.7-32), rgdal
Description Functions for computing geographically weighted regressions are provided, based on work by Chris Brunsdon, Martin Charlton and Stewart Fotheringham.
License GPL (>= 2)
RoxygenNote 6.1.1
NeedsCompilation yes
Author Roger Bivand [cre, aut] (<https://orcid.org/0000-0003-2392-6140>),
Danlin Yu [aut],
Tomoki Nakaya [ctb],
Miquel-Angel Garcia-Lopez [ctb]
Maintainer Roger Bivand <Roger.Bivand@nhh.no>
Repository CRAN
Date/Publication 2020-02-11 08:40:02 UTC

R topics documented:

georgia ....................................................... 2
ggwr .......................................................... 3
ggwr.sel ..................................................... 5
gw.adapt .................................................... 6
gw.cov ....................................................... 7
gwr .......................................................... 9
gwr.bisquare ............................................... 13
gwr.gauss .................................................. 14
gwr.morantest ............................................. 15
The Georgia census data set from Fotheringham et al. (2002) in shapefile format.

Usage
data(georgia)

Format
A SpatialPolygonsDataFrame object (proj4string set to "+proj=longlat +datum=NAD27").
The "data" slot is a data frame with 159 observations on the following 13 variables.

AreaKey  a numeric vector
Latitude  a numeric vector
Longitude a numeric vector
TotPop90  a numeric vector
PctRural  a numeric vector
PctBach  a numeric vector
PctEld  a numeric vector
PctFB  a numeric vector
PctPov  a numeric vector
PctBlack  a numeric vector
ID  a numeric vector
X  a numeric vector
Y  a numeric vector

Details
Variables are from GWR3 file GeorgiaData.csv.

Source
Originally: http://www.census.gov/geo/cob/bdy/co/co90shp/co13_d90_shp.zip, currently behind:
References


Examples

data(georgia)
plot(gSRDF)
data(gSRouter)

Description

The function implements generalised geographically weighted regression approach to exploring spatial non-stationarity for given global bandwidth and chosen weighting scheme.

Usage

ggwr(formula, data = list(), coords, bandwidth, gweight = gwr.Gauss, adapt = NULL, fit.points, family = gaussian, longlat = NULL, type = c("working", "deviance", "pearson", "response"))

Arguments

- formula: regression model formula as in glm
- data: model data frame as in glm, or may be a SpatialPointsDataFrame or SpatialPolygonsDataFrame object as defined in package sp
- coords: matrix of coordinates of points representing the spatial positions of the observations
- bandwidth: bandwidth used in the weighting function, possibly calculated by ggwr.sel
- gweight: geographical weighting function, at present gwr.Gauss() default, or gwr.biSquare(), the previous default or gwr.bisquare()
- adapt: either NULL (default) or a proportion between 0 and 1 of observations to include in weighting scheme (k-nearest neighbours)
- fit.points: an object containing the coordinates of fit points; often an object from package sp; if missing, the coordinates given through the data argument object, or the coords argument are used
- family: a description of the error distribution and link function to be used in the model, see glm
- longlat: TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself
- type: the type of residuals which should be returned. The alternatives are: "working" (default), "pearson", "deviance" and "response"
Value

A list of class “gwr”:

- **SDF**
  a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") with fit.points, weights, GWR coefficient estimates, dispersion if a "quasi"-family is used, and the residuals of type "type" in its "data" slot.

- **lhat**
  Leung et al. L matrix, here set to NA

- **lm**
  GLM global regression on the same model formula.

- **bandwidth**
  the bandwidth used.

- **this.call**
  the function call used.

Note

The use of GWR on GLM is only at the initial proof of concept stage, nothing should be treated as an accepted method at this stage.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

ggwr.sel, gwr

Examples

```r
if (require(rgdal)) {
  xx <- readOGR(system.file("shapes/sids.shp", package="spData")[[1]])
bw <- 144.4813
  ## Not run:
bw <- ggwr.sel(SID74 ~ I(NWBIR74/BIR74) + offset(log(BIR74)), data=xx,
                   family=poisson(), longlat=TRUE)
  ## End(Not run)
nc <- ggwr(SID74 ~ I(NWBIR74/BIR74) + offset(log(BIR74)), data=xx,
           family=poisson(), longlat=TRUE, bandwidth=bw)
  ## Not run:
nc <- ggwr(SID74 ~ I(NWBIR74/10000) + offset(log(BIR74)), data=xx,
           family=poisson(), longlat=TRUE, bandwidth=bw)
nc
nc <- ggwr(SID74 ~ I(NWBIR74/10000) + offset(log(BIR74)), data=xx,
           family=quasipoisson(), longlat=TRUE, bandwidth=bw)
nc
```
ggwr.sel

Crossvalidation of bandwidth for generalised GWR

Description

The function finds a bandwidth for a given generalised geographically weighted regression by optimizing a selected function. For cross-validation, this scores the root mean square prediction error for the generalised geographically weighted regressions, choosing the bandwidth minimizing this quantity.

Usage

```r
ggwr.sel(formula, data = list(), coords, adapt = FALSE, gweight = gwr.Gauss,
  family = gaussian, verbose = TRUE, longlat = NULL, RMSE=FALSE,
  tol=.Machine$double.eps^0.25)
```

Arguments

- **formula**: regression model formula as in `glm`
- **data**: model data frame as in `glm`, or may be a SpatialPointsDataFrame or SpatialPolygonsDataFrame object as defined in package `sp`
- **coords**: matrix of coordinates of points representing the spatial positions of the observations
- **adapt**: either TRUE: find the proportion between 0 and 1 of observations to include in weighting scheme (k-nearest neighbours), or FALSE — find global bandwidth
- **gweight**: geographical weighting function, at present `gwr.Gauss()` default, or `gwr.gauss()`, the previous default or `gwr.bisquare()`
- **family**: a description of the error distribution and link function to be used in the model, see `glm`
- **verbose**: if TRUE (default), reports the progress of search for bandwidth
- **longlat**: TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if `x` is a SpatialPoints object, the value is taken from the object itself
- **RMSE**: default FALSE to correspond with CV scores in newer references (sum of squared CV errors), if TRUE the previous behaviour of scoring by LOO CV RMSE
- **tol**: the desired accuracy to be passed to optimize

Value

returns the cross-validation bandwidth.
Note
The use of GWR on GLM is only at the initial proof of concept stage, nothing should be treated as an accepted method at this stage.

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
gwr.sel, ggwr

Examples

```r
if (require(rgdal)) {
  xx <- readOGR(system.file("shapes/sids.shp", package="spData")[[1]])
  bw <- gwr.sel(SID74 ~ I(NWBIR74/BIR74) + offset(log(BIR74)), data=xx, 
    family=poisson(), longlat=TRUE)
  bw
}
```

---

**gw.adapt**

Adaptive kernel for GWR

**Description**

The function constructs weights using an adaptive kernel for geographically weighted regression

**Usage**

```r
gw.adapt(dp, fp, quant, longlat=FALSE)
```

**Arguments**

- **dp**: data points coordinates
- **fp**: fit points coordinates
- **quant**: proportion of data points to include in the weights
- **longlat**: if TRUE, use distances on an ellipse with WGS84 parameters
**gw.cov**

**Value**

a vector of weights

**Author(s)**

Roger Bivand <Roger.Bivand@nhh.no>

---

**gw.cov**

*Geographically weighted local statistics*

**Description**

The function provides an implementation of geographically weighted local statistics based on Chapter 7 of the GWR book - see references. Local means, local standard deviations, local standard errors of the mean, standardised differences of the global and local means, and local covariances and if requested correlations, are reported for the chosen fixed or adaptive bandwidth and weighting function.

**Usage**

```r
gw.cov(x, vars, fp, adapt = NULL, bw, gweight = gwr.bisquare, cor = TRUE, var.term = FALSE, longlat = NULL)
```

**Arguments**

- `x` should be a SpatialPolygonsDataFrame object or a SpatialPointsDataFrame object
- `vars` is a vector of column names of the data frame in the data slot of `x`
- `fp` if given an object inheriting from “Spatial” that contains the fit points to be used, for example a SpatialPixels object describing the grid of points to be used
- `adapt` if given should lie between 0 and 1, and indicates the proportion of observations to be included in the weighted window - it cannot be selected automatically
- `bw` when adapt is not given, the bandwidth chosen to suit the data set - it cannot be selected automatically
- `gweight` default gwr.bisquare - the weighting function to use
- `cor` default TRUE, report correlations in addition to covariances
- `var.term` default FALSE, if TRUE apply a correction to the variance term
- `longlat` TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if `x` is a SpatialPoints object, the value is taken from the object itself
Value

If argument fp is given, and it is a SpatialPixels object, a SpatialPixelsDataFrame is returned, if it is any other coordinate object, a SpatialPointsDataFrame is returned. If argument fp is not given, the object returned will be the class of object x. The data slot will contain a data frame with local means, local standard deviations, local standard errors of the mean, standardised differences of the global and local means, and local covariances and if requested correlations.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

gwr

Examples

data(georgia)
SRgwls <- gw.cov(gSRDF, vars=6:11, bw=2, longlat=FALSE)
names(SRgwls$SDF)
spplot(SRgwls$SDF, "mean.PctPov")
spplot(SRgwls$SDF, "sd.PctPov")
spplot(SRgwls$SDF, "sem.PctPov")
spplot(SRgwls$SDF, "diff.PctPov")
"
SRgwls <- gw.cov(gSRDF, vars=6:11, bw=150, longlat=TRUE)
names(SRgwls$SDF)
spplot(SRgwls$SDF, "mean.PctPov")
spplot(SRgwls$SDF, "sd.PctPov")
spplot(SRgwls$SDF, "sem.PctPov")
spplot(SRgwls$SDF, "diff.PctPov")
"
SRgwls <- gw.cov(gSRDF, vars=6:11, fp=gGrid, bw=150, longlat=TRUE)
names(SRGgwls$SDF)
spplot(SRGgwls$SDF, "mean.PctPov")
spplot(SRGgwls$SDF, "sd.PctPov")
spplot(SRGgwls$SDF, "sem.PctPov")
spplot(SRGgwls$SDF, "diff.PctPov")
"
#gGrid <- sample.Polygons(slot(gSRouter, "polygons")[[1]], 5000,
gGrid <- spsample(slot(gSRouter, "polygons")[[1]], 5000,
type="regular")
gridded(gGrid) <- TRUE
SGgwls <- gw.cov(gSRDF, vars=6:11, fp=gGrid, bw=150, longlat=TRUE)
names(SGgwls$SDF)
spplot(SGgwls$SDF, "mean.PctPov")
spplot(SGgwls$SDF, "sd.PctPov")
spplot(SGgwls$SDF, "sem.PctPov")
spplot(SGgwls$SDF, "diff.PctPov")
"
set.seed(1)
pts <- data.frame(x=runif(100, 0, 5), y=runif(100, 0, 5), z=rnorm(100))
coordinates(pts) <- c("x", "y")
proj4string(pts) <- CRS("+proj=longlat +ellps=WGS84")
fps <- SpatialPoints(cbind(x=runif(100, 0, 5), y=runif(100, 0, 5)),
                      proj4string=CRS("+proj=longlat +ellps=WGS84"))
t0 <- gw.cov(pts, "z", fp=fps, adapt=0.2, longlat=TRUE)

---

**gwr**

*Geographically weighted regression*

**Description**

The function implements the basic geographically weighted regression approach to exploring spatial non-stationarity for given global bandwidth and chosen weighting scheme.

**Usage**

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

**Arguments**

- `formula`: regression model formula as in `lm`
- `data`: model data frame, or SpatialPointsDataFrame or SpatialPolygonsDataFrame as defined in package *sp*
- `coords`: matrix of coordinates of points representing the spatial positions of the observations; may be omitted if the object passed through the data argument is from package *sp*
- `bandwidth`: bandwidth used in the weighting function, possibly calculated by `gwr.sel`
- `gweight`: geographical weighting function, at present `gwr.Gauss()` default, or `gwr.gauss()`, the previous default or `gwr.bisquare()`
- `adapt`: either NULL (default) or a proportion between 0 and 1 of observations to include in weighting scheme (k-nearest neighbours)
- `hatmatrix`: if TRUE, return the hatmatrix as a component of the result, ignored if `fit.points` given
- `fit.points`: an object containing the coordinates of fit points; often an object from package *sp*; if missing, the coordinates given through the data argument object, or the coords argument are used
- `longlat`: TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself
se.fit  if TRUE, return local coefficient standard errors - if hatmatrix is TRUE and no fit.points are given, two effective degrees of freedom sigmas will be used to generate alternative coefficient standard errors
weights  case weights used as in weighted least squares, beware of scaling issues, probably unsafe
cl  if NULL, ignored, otherwise cl must be an object describing a "cluster" created using makeCluster in the parallel package. The cluster will then be used to hand off the calculation of local coefficients to cluster nodes, if fit points have been given as an argument, and hatmatrix=FALSE
predictions  default FALSE; if TRUE and no fit points given, return GW fitted values at data points, if fit points given and are a Spatial*DataFrame object containing the RHS variables in the formula, return GW predictions at the fit points
fittedGWRobject  a fitted gwr object with a hatmatrix (optional), if given, and if fit.points are given and if se.fit is TRUE, two effective degrees of freedom sigmas will be used to generate alternative coefficient standard errors
se.fit.CCT  default TRUE, compute local coefficient standard errors using formula (2.14), p. 55, in the GWR book
x  an object of class "gwr" returned by the gwr function
...  arguments to be passed to other functions

Details
The function applies the weighting function in turn to each of the observations, or fit points if given, calculating a weighted regression for each point. The results may be explored to see if coefficient values vary over space. The local coefficient estimates may be made on a multi-node cluster using the cl argument to pass through a parallel cluster. The function will then divide the fit points (which must be given separately) between the clusters for fitting. Note that each node will need to have the "spgwr" package present, so initiating by clusterEvalQ(cl,library(spgwr)) may save a little time per node. The function clears the global environment on the node of objects sent. Using two nodes reduces timings to a little over half the time for a single node.

The section of the examples code now includes two simulation scenarios, showing how important it is to check that mapped pattern in local coefficients is actually there, rather than being an artefact.

Value
A list of class "gwr":

SDF  a SpatialPointsDataFrame (may be gridded) or SpatialPolygonsDataFrame object (see package "sp") with fit.points, weights, GWR coefficient estimates, R-squared, and coefficient standard errors in its "data" slot.
1hat  Leung et al. L matrix
lm  Ordinary least squares global regression on the same model formula, as returned by lm.wfit().
bandwidth  the bandwidth used.
this.call  the function call used.
Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
gwr.sel, gwr.gauss, gwr.bisquare

Examples
data(columbus, package="spData")
col.lm <- lm(CRIME ~ INC + HOVAL, data=columbus)
summary(col.lm)
col.bw <- gwr.sel(CRIME ~ INC + HOVAL, data=columbus,
   coords=cbind(columbus$x, columbus$y))
col.gauss <- gwr(CRIME ~ INC + HOVAL, data=columbus,
   coords=cbind(columbus$x, columbus$y), bandwidth=col.bw, hatmatrix=TRUE)
col.gauss
col.d <- gwr.sel(CRIME ~ INC + HOVAL, data=columbus,
   coords=cbind(columbus$x, columbus$y), gweight=gwr.bisquare)
col.bisq <- gwr(CRIME ~ INC + HOVAL, data=columbus,
   coords=cbind(columbus$x, columbus$y), bandwidth=col.d,
   gweight=gwr.bisquare, hatmatrix=TRUE)
col.bisq
data(georgia)
g.adapt.gauss <- gwr.sel(PctBach ~ TotPop90 + PctRural + PctEld + PctFB +
PctPov + PctBlack, data=gSRDF, adapt=TRUE)
res.adpt <- gwr(PctBach ~ TotPop90 + PctRural + PctEld + PctFB + PctPov + PctBlack, data=gSRDF, adapt=g.adapt.gauss)
res.adpt
pairs(as(res.adpt$SDF, "data.frame")[,2:8], pch=".")
brks <- c(-0.25, 0, 0.01, 0.025, 0.075)
cols <- grey(5:2/6)
plot(res.adpt$SDF, col=cols[findInterval(res.adpt$SDF$PctBlack, brks,
   all.inside=TRUE)])

# simulation scenario with patterned dependent variable
set.seed(1)
X0 <- runif(nrow(gSRDF)*3)
X1 <- matrix(sample(X0), ncol=3)
X1 <- prcomp(X1, center=FALSE, scale.=FALSE)$x
gSRDF$X1 <- X1[,1]
gSRDF$X2 <- X1[,2]
gSRDF$X3 <- X1[,3]
bw <- gwr.sel(PctBach ~ X1 + X2 + X3, data=gSRDF, verbose=FALSE)
out <- gwr(PctBach ~ X1 + X2 + X3, data=gSRDF, bandwidth=bw, hatmatrix=TRUE)
out
spplot(gSRDF, "PctBach", col.regions=grey.colors(20))
spplot(gSRDF, c("X1", "X2", "X3"), col.regions=grey.colors(20)) # pattern in the local coefficients
spplot(out$SDF, c("X1", "X2", "X3"), col.regions=grey.colors(20)) # but no "significant" pattern
spplot(out$SDF, c("X1_se", "X2_se", "X3_se"), col.regions=grey.colors(20))
out$SDF$X1_t <- out$SDF$X1/out$SDF$X1_se
out$SDF$X2_t <- out$SDF$X2/out$SDF$X2_se
out$SDF$X3_t <- out$SDF$X3/out$SDF$X3_se
spplot(out$SDF, c("X1_t", "X2_t", "X3_t"), col.regions=grey.colors(20)) # simulation scenario with random dependent variable
yrn <- rnorm(nrow(gSRDF))
gSRDF$yrn <- sample(yrn)
# bandwidth selection maxes out at 620 km, equal to upper bound
bw <- gwr.sel(yrn ~ X1 + X2 + X3, data=gSRDF, verbose=FALSE)
out <- gwr(yrn ~ X1 + X2 + X3, data=gSRDF, bandwidth=bw, hatmatrix=TRUE)

# end of simulations

data(meuse)
coordinates(meuse) <- c("x", "y")
meuse$ffreq <- factor(meuse$ffreq)
data(meuse.grid)
coordinates(meuse.grid) <- c("x", "y")
meuse.grid$ffreq <- factor(meuse.grid$ffreq)
gridded(meuse.grid) <- TRUE
xx <- gwr(cadmium ~ dist, meuse, bandwidth = 228, hatmatrix=TRUE)
xx
x <- gwr(cadmium ~ dist, meuse, bandwidth = 228, fit.points = meuse.grid, predict=TRUE, se.fit=TRUE, fittedGWRobject=xx)
x
spplot(x$SDF, "pred")
spplot(x$SDF, "pred.se")

## Not run:
g.bw.gauss <- gwr.sel(PctBach ~ TotPop90 + PctRural + PctEld + PctFB + PctPov + PctBlack, data=gSRDF)
res.bw <- gwr(PctBach ~ TotPop90 + PctRural + PctEld + PctFB + PctPov + PctBlack, data=gSRDF, bandwidth=g.bw.gauss)
res.bw
pairs(as(res.bw$SDF, "data.frame")[,2:8], pch=".")
plot(res.bw$SDF, col=cols[findInterval(res.bw$SDF$PctBlack, brks, all.inside=TRUE)])
g.bw.gauss <- gwr.sel(PctBach ~ TotPop90 + PctRural + PctEld + PctFB +
                      PctPov + PctBlack, data=gSRDF, longlat=TRUE)
data(gSRouter)
require(maptools)
SG <- GE_SpatialGrid(gSRouter, maxPixels = 100)
SPxMASK0 <- over(SG$SG, gSRouter)
SGDF <- SpatialGridDataFrame(slot(SG$SG, "grid"),
                           data=data.frame(SPxMASK0=SPxMASK0),
                           proj4string=CRS(proj4string(gSRouter))))
SPxDF <- as(SGDF, "SpatialPixelsDataFrame")
res.bw <- gwr(PctBach ~ TotPop90 + PctRural + PctEld + PctFB + PctPov +
              PctBlack, data=gSRDF, bandwidth=g.bw.gauss, fit.points=SPxDF,
              longlat=TRUE)
res.bw
res.bw$timings
spplot(res.bw$SDF, "PctBlack")
require(parallel)
c1 <- makeCluster(detectCores())
res.bwc <- gwr(PctBach ~ TotPop90 + PctRural + PctEld + PctFB + PctPov +
              PctBlack, data=gSRDF, bandwidth=g.bw.gauss, fit.points=SPxDF,
              longlat=TRUE, cl=c1)
res.bwc
res.bwc$timings
stopCluster(c1)

# End(Not run)

gwr.bisquare

GWR bisquare weights function

Description

The function returns a vector of weights using the bisquare scheme:

\[ w_{ij}(g) = \left(1 - \left(\frac{d_{ij}^2}{d^2}\right)\right)^2 \]

if \( d_{ij} \leq d \) else \( w_{ij}(g) = 0 \), where \( d_{ij} \) are the distances between the observations and \( d \) is the distance at which weights are set to zero.

Usage

gwr.bisquare(dist2, d)

Arguments

dist2 vector of squared distances between observations
d distance at which weights are set to zero
gwr.gauss

Value
matrix of weights.

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
gwr.sel, gwr

Examples
plot(seq(-10,10,0.1), gwr.bisquare(seq(-10,10,0.1)^2, 6.0), type="l")

Description
The gwr.gauss function returns a vector of weights using the Gaussian scheme:
\[ w(g) = e^{-\frac{(d/h)^2}{2}} \]
where \( d \) are the distances between the observations and \( h \) is the bandwidth.
The default (from release 0.5) gwr.Gauss function returns a vector of weights using the Gaussian scheme:
\[ w(g) = e^{-\frac{1}{2}(d/h)^2} \]

Usage
gwr.gauss(dist2, bandwidth)
gwr.Gauss(dist2, bandwidth)

Arguments
dist2 vector of squared distances between observations and fit point
bandwidth bandwidth
Value

vector of weights.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no>

References


See Also

gwr.sel, gwr

Examples

```r
plot(seq(-10,10,0.1), gwr.Gauss(seq(-10,10,0.1)^2, 3.5), type="l")
```

Description

The function returns Leung et al. (2000) three moment approximation for Moran’s I, for a gwr object calculated with argument hatmatrix=TRUE. This implementation should not be regarded as authoritative, as it involves assumptions about implied methods and about estimated degrees of freedom.

Usage

gwr.morantest(x, lw, zero.policy = FALSE)

Arguments

- **x**: a gwr object returned by `gwr()` with argument hatmatrix=TRUE
- **lw**: a listw object created for example by `nb2listw` in the spdep package
- **zero.policy**: if TRUE assign zero to the lagged value of zones without neighbours, if FALSE (default) assign NA

Value

a “htest” object with the results of testing the GWR residuals
Author(s)
Roger Bivand

References

Examples
if (suppressWarnings(require(spData)) && suppressWarnings(require(spdep))) {
  data(columbus, package="spData")
  bw <- gwr.sel(CRIME ~ INC + HOVAL, data=columbus, coords=coords)
  col0 <- gwr(CRIME ~ INC + HOVAL, data=columbus, coords=coords,
              bandwidth=bw, hatmatrix=TRUE)
  gwr.morantest(col0, nb2listw(col.gal.nb))
}

---

gwr.sel

**Crossvalidation of bandwidth for geographically weighted regression**

Description
The function finds a bandwidth for a given geographically weighted regression by optimizing a selected function. For cross-validation, this scores the root mean square prediction error for the geographically weighted regressions, choosing the bandwidth minimizing this quantity.

Usage
gwr.sel(formula, data=list(), coords, adapt=FALSE, gweight=gwr.Gauss,
        method = "cv", verbose = TRUE, longlat=NULL, RMSE=FALSE, weights,
        tol=.Machine$double.eps^0.25, show.error.messages = FALSE)

Arguments
- **formula**: regression model formula as in `lm`
- **data**: model data frame as in `lm`, or may be a `SpatialPointsDataFrame` or `SpatialPolygonsDataFrame` object as defined in package `sp`
- **coords**: matrix of coordinates of points representing the spatial positions of the observations
- **adapt**: either TRUE: find the proportion between 0 and 1 of observations to include in weighting scheme (k-nearest neighbours), or FALSE — find global bandwidth
- **gweight**: geographical weighting function, at present `gwr.Gauss()` default, or `gwr.gauss()`, the previous default or `gwr.bisquare()`
- **method**: default "cv" for drop-1 cross-validation, or "aic" for AIC optimisation (depends on assumptions about AIC degrees of freedom)
verbose if TRUE (default), reports the progress of search for bandwidth
longlat TRUE if point coordinates are longitude-latitude decimal degrees, in which case distances are measured in kilometers; if x is a SpatialPoints object, the value is taken from the object itself
RMSE default FALSE to correspond with CV scores in newer references (sum of squared CV errors), if TRUE the previous behaviour of scoring by LOO CV RMSE
weights case weights used as in weighted least squares, beware of scaling issues — only used with the cross-validation method, probably unsafe
tol the desired accuracy to be passed to optimize
show.error.messages default FALSE; may be set to TRUE to see error messages if gwr.sel returns without a value

Details
If the regression contains little pattern, the bandwidth will converge to the upper bound of the line search, which is the diagonal of the bounding box of the data point coordinates for “adapt=FALSE”, and 1 for “adapt=TRUE”; see the simulation block in the examples below.

Value
returns the cross-validation bandwidth.

Note
Use of method="aic" results in the creation of an n by n matrix, and should not be chosen when n is large.

Author(s)
Roger Bivand <Roger.Bivand@nhh.no>

References

See Also
gwr.bisquare, gwr.gauss

Examples
data(columbus, package="spData")
gwr.sel(CRIME ~ INC + HOVAL, data=columbus, coords=cbind(columbus$X, columbus$Y))
gwr.sel(CRIME ~ INC + HOVAL, data=columbus,
coords=cbind(columbus$X, columbus$Y), gweight=gwr.bisquare)
## Not run:
data(georgia)
set.seed(1)
X0 <- runif(nrow(gSRDF)*3)
X1 <- matrix(sample(X0), ncol=3)
X1 <- prcomp(X1, center=FALSE, scale.=FALSE)$x
gSRDF$X1 <- X1[,1]
gSRDF$X2 <- X1[,2]
gSRDF$X3 <- X1[,3]
yrn <- rnorm(nrow(gSRDF))
gSRDF$yrn <- sample(yrn)
bw <- gwr.sel(yrn ~ X1 + X2 + X3, data=gSRDF, method="cv", adapt=FALSE, verbose=FALSE)
bw
bw <- gwr.sel(yrn ~ X1 + X2 + X3, data=gSRDF, method="aic", adapt=FALSE, verbose=FALSE)
bw
bw <- gwr.sel(yrn ~ X1 + X2 + X3, data=gSRDF, method="cv", adapt=TRUE, verbose=FALSE)
bw
bw
bw <- gwr.sel(yrn ~ X1 + X2 + X3, data=gSRDF, method="aic", adapt=TRUE, verbose=FALSE)
bw
## End(Not run)

gwr.tricube

GWR tricube weights function

Description

The function returns a vector of weights using the tricube scheme:

\[ w_{ij}(g) = \left(1 - \left(\frac{d_{ij}}{d}\right)^3\right)^3 \]

if \( d_{ij} \leq d \) else \( w_{ij}(g) = 0 \), where \( d_{ij} \) are the distances between the observations and \( d \) is the distance at which weights are set to zero.

Usage

gwr.tricube(dist2, d)

Arguments

dist2 vector of squared distances between observations
d distance at which weights are set to zero

Value

matrix of weights.
**Description**

Four related test statistics for comparing OLS and GWR models based on papers by Brunsdon, Fotheringham and Charlton (1999) and Leung et al (2000), and a development from the GWR book (2002).

**Usage**

- `LMZ.F3GWR.test()`
- `LMZ.F2GWR.test(x)`
- `LMZ.F1GWR.test(x)`
- `BFC99.gwr.test(x)`
- `BFC02.gwr.test(x, approx=FALSE)`
- `anova(object, ..., approx=FALSE)`

**Arguments**

- `go, x, object`: A gwr object returned by gwr()
- `...`: Arguments passed through (unused)
- `approx`: Default FALSE, if TRUE, use only (n - tr(S)) instead of (n - 2*tr(S) - tr(S'S)) as the GWR degrees of freedom

**Details**

The papers in the references give the background for the analyses of variance presented.
Value

BFC99.GWR.test, BFC02.gwr.test, LMZ.F1GWR.test and LMZ.F2GWR.test return "htest" objects, LMZ.F3GWR.test a matrix of test results.

Author(s)

Roger Bivand <Roger.Bivand@nhh.no> and Danlin Yu

References


See Also

gwr

Examples

data(columbus, package="spData")
col.bw <- gwr.sel(CRIME ~ INC + HOVAL, data=columbus,
coords=cbind(columbus$X, columbus$Y))
col.gauss <- gwr(CRIME ~ INC + HOVAL, data=columbus,
coords=cbind(columbus$X, columbus$Y), bandwidth=col.bw, hatmatrix=TRUE)
BFC99.gwr.test(col.gauss)
BFC02.gwr.test(col.gauss)
BFC02.gwr.test(col.gauss, approx=TRUE)
anova(col.gauss)
anova(col.gauss, approx=TRUE)
## Not run:
col.d <- gwr.sel(CRIME ~ INC + HOVAL, data=columbus,
coords=cbind(columbus$X, columbus$Y), gweight=gwr.bisquare)
col.bisq <- gwr(CRIME ~ INC + HOVAL, data=columbus,
coords=cbind(columbus$X, columbus$Y), bandwidth=col.d,
gweight=gwr.bisquare, hatmatrix=TRUE)
BFC99.gwr.test(col.bisq)
## End(Not run)
Index

*Topic **datasets**
  * georgia, 2
*Topic **spatial**
  * ggwr, 3
  * ggwr.sel, 5
  * gw.adapt, 6
  * gw.cov, 7
  * gwr, 9
  * gwr.bisquare, 13
  * gwr.gauss, 14
  * gwr.morantest, 15
  * gwr.sel, 16
  * gwr.tricube, 18
  * LMZ.F3GWR.test, 19

anova.gwr (LMZ.F3GWR.test), 19
BFC02.gwr.test (LMZ.F3GWR.test), 19
BFC99.gwr.test (LMZ.F3GWR.test), 19

georgia, 2
ggwr, 3, 6
ggwr.cv.adapt.f (ggwr.sel), 5
ggwr.cv.f (ggwr.sel), 5
ggwr.sel, 4, 5
gSRDF (georgia), 2
gSRouter (georgia), 2
gw.adapt, 6
gw.cov, 7
gwr, 4, 8, 9, 14, 15, 19, 20
gwr.aic.adapt.f (gwr.sel), 16
gwr.aic.f (gwr.sel), 16
gwr.bisquare, 11, 13, 17
gwr.cv.adapt.f (gwr.sel), 16
gwr.cv.f (gwr.sel), 16
gwr.Gauss (gwr.gauss), 14
gwr.gauss, 11, 14, 17
gwr.morantest, 15
gwr.sel, 6, 11, 14, 15, 16, 19
gwr.tricube, 18