Package 'McSpatial'

February 19, 2015

Type Package

Title Nonparametric spatial data analysis

Version 2.0

Date 2013-5-20

Author Daniel McMillen

Maintainer Daniel McMillen <mcmillen@illinois.edu>

Description Locally weighted regression, semiparametric and conditionally parametric regression, fourier and cubic spline functions, GMM and linearized spatial logit and probit, k-density functions and counterfactuals, nonparametric quantile regression and conditional density functions, Machado-Mata decomposition for quantile regressions, spatial AR model, repeat sales models, conditionally parametric logit and probit

License GPL

LazyLoad yes

Depends lattice, locfit, maptools, quantreg, RANN, SparseM

Suggests car, classInt, mlogit, RColorBrewer, spatstat, spdep

NeedsCompilation no

Repository CRAN

Date/Publication 2013-05-26 11:08:53

R topics documented:

McSpatial-package *Nonparametric spatial data analysis*

Description

Locally weighted regression, semiparametric and conditionally parametric regression, fourier and cubic spline functions, GMM and linearized spatial logit and probit, k-density functions and counterfactuals, nonparametric quantile regression and conditional density functions, Machado-Mata decomposition for quantile regressions, spatial AR model, repeat sales models, conditionally parametric logit and probit

Details

Package:	McSpatial
Type:	Package
Version:	2.0
Date:	2013-5-20
License:	GPL
LazyLoad:	yes

Author(s)

Daniel McMillen

Maintainer: Daniel McMillen <mcmillen@illinois.edu>

condens *Conditional density estimation*

Description

Estimates conditional density functions of the form f(y| x) = f(x,y)/f(x). Kernel density estimators are used to estimate f(x,y) and f(x). The conditional density function can be plotted as a three-dimensional surface or as a contour map. Alternatively, the conditional density of y can be graphed for as many as five target values of x.

Usage

```
condens(form,window=.7,bandwidth=0,kern="tcub",
  mingrid.x=NULL,maxgrid.x=NULL,mingrid.y=NULL,maxgrid.y=NULL,ngrid=50,
  xlab="x",ylab="y",zlab="fxy/fx",contour=TRUE,level=TRUE,wire=TRUE,dens=TRUE,
  targetx.dens=NULL,quantile.dens=c(.10,.25,.50,.75,.90),data=NULL)
```

Arguments

`form`	Model formula
`window`	Window size. Default: 0.25.
`bandwidth`	Bandwidth. Default: not used.
`kern`	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
`mingrid.x, maxgrid.x, mingrid.y, maxgrid.y, ngrid`	The mingrid and maxgrid values are the boundaries for the *ngrid x ngrid* lattice used in the graphs produced by *condens*. By default, mingrid.x = min(x), maxgrid.x = max(x), mingrid.y = min(y), maxgrid.y = max(y), and ngrid=50.
`xlab`	Label for the *x*-axis in graphs. Default: "x"
`ylab`	Label for the *y*-axis in graphs. Default: "y"
`zlab`	Label for the *z*-axis in graphs. Default: "fxy/fx"
`contour`	If *contour=T*, produces a two-dimensional contour plot of the conditional density estimates. Evaluated for an *ngrid* x *ngrid* lattice. Default is *contour=T*.
`level`	If *level=T*, produces a two-dimensional level plot of the conditional density estimates. Evaluated for an *ngrid* x *ngrid* lattice. Default is *level=F*.
`wire`	If *wire=T*, produces a three-dimensional plot of the conditional density estimates. Evaluated for an *ngrid* x *ngrid* lattice. Default is *wire=T*.
`dens`	If *dens=T*, produces a plot showing how *f(y|x)* varies over *y* for given target values of *x*. Target values of *x* are provided using the *targetx.dens* or *quantile.dens* options. Default is dens=F.
`targetx.dens`	Target values for *x* in the density plots, e.g, *targetx.dens* = c(200,400,600). Maximum number of entries is 5. If *targetx.dens* has more than 5 entries, only the first 5 will be used. Default is *targetx.dens = NULL*, meaning that the target values for *x* are determined by the *quantile.dens* option.
`quantile.dens`	Quantiles for the target values for *x* in the density plots, e.g, *quantile.dens* = c(.25,.50,.75). Maximum number of entries is 5. If quantile.dens has more than 5 entries, only the first 5 will be used. Default is *quantile.dens* = c(.10,.25,.50,.75,.90).
`data`	A data frame containing the data. Default: use data in the current working directory.

Details

The *locfit* package is used to find the target values of *x* for *f(x)* and *y* for *f(y)*. The *expand.grid* command is then used to determine the target values of *x* and *y* for *f(x,y)*. The *smooth12* command is used to interpolate *f(x)*, *f(y)*, and *f(x,y)* to the full data set and to the grid of target values for the contour, level, and wire plots.

The density functions *f(x)* and *f(y)* are as follows:

$$f(x) = \frac{1}{sd(x) * b * n} \sum_i K(\frac{x_i - x}{sd(x) * b})$$

$$f(y) = \frac{1}{sd(y) * b * n} \sum_i K(\frac{y_i - y}{sd(y) * b})$$

A product kernel is used for *f(x,y)*:

$$f(x,y) = \frac{1}{sd(x) * b * sd(y) * b * n} \sum_i K(\frac{x_i - x}{sd(x) * b}) K(\frac{y_i - y}{sd(y) * b})$$

where *b* is the bandwidth and the target points are *x* and *y*. The bandwidth, *b*, can be set using the *bandwidth* option. If *b* = 0 (the default), *sd(x)*b* and *sd(y)*b* are replaced by window values, $h = quantile(dist, window)$, where $dist = |x_i - x|$ or $dist = |y_i - y|$. The window size is set using the *window* option. By default, *window* = .7 and *bandwidth* = 0. Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2} I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|) I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5} e^{-z^2/2}$

The contour, level, and wire plots are produced from the values in *gridmat* using the *lattice* package. The two-dimensional density graphs produced when *dens=TRUE* are plots of *f(y,x)/f(x)* at given values of *x*. By default, the values for *x* are the quantiles given in *quantile.dens*. Alternatively, the values of *x* can be specified directly using the *targetx.dens* option. The values used to construct the density graphs are stored in *densmat*. Both *gridmat* and *densmat* are stored by *condens* even if the printing of the graphs is suppressed.

Value

`fx` The values of *f(x)*, one for each data point.

`fy` The values of *f(y)*, one for each data point.

`fxy` The values of *f(x,y)*, one for each data point. The conditional densities are fxy/fx for *x* and fxy/fy for *y*.

`gridmat` An (*ngrid*ngrid*)x3 matrix used to produce the contour, level, and wire maps. The first column contains the lattice values for *x*, the second column contains the lattice values for *y*, and the third column has the estimated values of *f(y|x)* at the target values for *x* and *y*.

`densmat` The estimated values of *f(y|x)* for the two-dimensional density graphs produced when *dens = TRUE*. If the number of observations in the call to *condens* is *n* and the number of entries in *quantile.dens* is *nq*, then densmat is an *n* x *nq* matrix.

References

Li, Oi and Jeffrey Scott Racine. *Nonparametric Econometrics: Theory and Practice.* Princeton, NJ: Princeton University Press, 2007.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics.* New York: Cambridge University Press, 1999.

See Also

qregcdf

Examples

```
data(dupage99)
dupage99$ratio <- dupage99$av/dupage99$price
dupage99$price <- dupage99$price/1000
par(ask=TRUE)
fit <- condens(ratio~price,contour=TRUE,level=TRUE,wire=TRUE,dens=TRUE,
  targetx.dens=seq(100,500,100), data=dupage99)
```

CookCensusTracts *Shapefile of Census Tracts in Cook County for 2000*

Description

A map of census tracts in Cook County, Illinois with data on population, median household income, average floor area ratios, and average home ages.

Format

A shape file with 1343 census tracts. Can be read directly into a GIS program.

`SP_ID` An identification variable

`AREA` Area of the tract in square miles

`TRACT` Census tract number, including state and county codes

`POPULATION` Population in 2000

`HHMEDINC` Median household income in the census tract in 2000

`FAR` Average floor area ratio for census tract in 2000. Calculated from Cook County assessment file

`AGE` Average age of homes in 2000. Calculated from Cook County assessment file.

`CHICAGO` Indicates whether the tract is part of the City of Chicago. O'Hare airport is NOT included in Chicago. It is identified in the variable *CAREA*.

`CAREA` For Chicago observations, indicates the community area. Missing for suburban observations.

Source

Daniel McMillen. Data are drawn from the U.S. Census and the Cook County Assessment File.

Examples

```
cook <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
sampvar <- cook$CHICAGO==1|(!is.na(cook$CAREA)&cook$CAREA=="O'Hare")
chicago <- cook[sampvar==TRUE,]
```

---

cookdata *Data set associated with CookCensusTracts shape file*

Description

Census tract data for Cook County, IL for 2000

Usage

```
data(cookdata)
```

Format

A data frame with 1343 observations on the following variables:

`AREA` Area of the census tract in square miles

`TRACT` Census tract number

`POPULATION` Total population

`HHMEDINC` Median household income

`FAR` Average floor-area ratio for single-family residential homes

`AGE` Average age of single-family residential homes

`CHICAGO` Indicates whether the tract is in Chicago. Note that the "O'Hare" community area is not counted as part of Chicago when defining this variable.

`CAREA` Community area name for tracts in Chicago

`LONGITUDE` Longitude of the tract centroid

`LATITUDE` Latitude of the tract centroid

`DCBD` Distance from the Chicago CBD, measured in miles

`LNFAR` Natural log of FAR

`LNDENS` Natural Log of Dens

See Also

CookCensusTracts

`cparlogit` *Conditionally Parametric logit for two or more choices*

Description

Estimates a logit model with two choices by maximizing a locally weighted likelihood function – the logit equivalent of cparlwr

Usage

```
cparlogit(form,nonpar,window=.25,bandwidth=0,kern="tcub",
distance="Mahal",target=NULL,data=NULL,minp=NULL)
```

Arguments

`form`	Model formula
`nonpar`	List of either one or two variables for *z*. Formats: *cparlogit(y~xlist, nonpar=~z1, ...)* or *cparlogit(y~xlist, nonpar=~z1+z2, ...)*. Important: note the "~" before the first *z* variable.
`window`	Window size. Default: 0.25.
`bandwidth`	Bandwidth. Default: not used.
`kern`	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
`distance`	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *cparlogit* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
`target`	If *target = NULL*, uses the *maketarget* command to form targets using the values specified for *window*, *bandwidth*, and *kern*. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
`data`	A data frame containing the data. Default: use data in the current working directory
`minp`	Specifies a limit for the estimated probability. Any estimated probability lower than *minp* will be set to *minp* and any probability higher than 1-*minp* will be set to 1-*minp*. By default, the estimated probabilities are bounded by 0 and 1.

Details

The list of explanatory variables is specified in the base model formula while *Z* is specified using *nonpar*. *X* can include any number of explanatory variables, but *Z* must have at most two.

The model is estimated by maximizing the following weighted log-likelihood function at each target point:

$$\sum_{i=1}^{n} w_i\{y_i log(P_i) + (1 - y_i)log(1 - P_i)\}$$

where y is the discrete dependent variable, X is the set of explanatory variables, and $P_i = \frac{exp(X_i\beta)}{1+exp(X_i\beta)}$.

When *Z* includes a single variable, w_i is a simple kernel weighting function: $w_i = K((z_i - z_0)/(sd(z) * h))$. When *Z* includes two variables (e.g., nonpar=~z1+z2), the method for specifying *w* depends on the *distance* option. Under either option, the *i*th row of the matrix *Z* = (z1, z2) is transformed such that $z_i = \sqrt{z_i * V * t(z_i)}$. Under the "Mahal" option, *V* is the inverse of cov(*Z*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*Z*)). After this transformation, the weights again reduce to the simple kernel weighting function $K((z_i - z_0)/(sd(z) * h))$. *h* is specified by the *bandwidth* or *window* option.

The great circle formula is used to construct the distances used to form the weights when *distance* = *"Latlong"*; in this case, the variable list for *nonpar* must be listed as *nonpar* = ~*latitude*+*longitude* (or ~*lo*+*la* or ~*lat*+*long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude are listed does not matter and the function only looks for the first two letters to determine which variable is latitude and which is longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance* = *"Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option.

Following White (1982), the covariance matrix for a quasi-maximum likelihood model is $A^{-1}BA^{-1}$, where

$$A = \sum_{i=1}^{n} w_i \frac{\partial^2 LnL_i}{\partial\beta\partial\beta'}$$

$$B = \sum_{i=1}^{n} w_i^2 \frac{\partial LnL_i}{\partial\beta} \frac{\partial LnL_i}{\partial\beta'}$$

For the logit model,

$$A = \sum_{i=1}^{n} w_i P_i(1 - P_i) X_i X_i'$$

$$B = \sum_{i=1}^{n} w_i^2 (y_i - P_i)^2 X_i X_i'$$

The covariance matrix is calculated at all target points and the implied standard errors are then interpolated to each data point.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|)I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$

Bi-Square	"bisq"	$\frac{15}{16}(1-z^2)^2 * I(\|z\|<1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1-\|z\|^3)^3 * I(\|z\|<1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1-z^2)^3 * I(\|z\|<1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target` The target points for the original estimation of the function.

`xcoef.target` Estimated coefficients, *B(z)*, at the target values of *z*.

`xcoef.target.se`
Standard errors for *B(z)* at the target values of *z*.

`xcoef` Estimated coefficients, *B(z)*, at the original data points.

`xcoef.se` Standard errors for *B(z)* with *z* evaluated at all points in the data set.

`p` The estimated probabilities.

`lnl` The log-likelihood value.

References

Fan, Jianqing, Nancy E. Heckman, and M.P. Wand, "Local Polynomial Kernel Regression for Generalized Linear Models and Quasi-Likelihood Functions," *Journal of the American Statistical Association* 90 (1995), 141-150.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

McMillen, Daniel P. and John F. McDonald, "Locally Weighted Maximum Likelihood Estimation: Monte Carlo Evidence and an Application," in Luc Anselin, Raymond J.G.M. Florax, and Sergio J. Rey, eds., *Advances in Spatial Econometrics*, Springer-Verlag, New York (2004), 225-239.

Tibshirani, Robert and Trevor Hastie, "Local Likelihood Estimation," *Journal of the American Statistical Association* 82 (1987), 559-568.

See Also

cparprobit

cparmlogit

gmmlogit

gmmprobit

splogit

spprobit

spprobitml

Examples

```
set.seed(5647)
data(cookdata)
cookdata <- cookdata[!is.na(cookdata$AGE),]
n = nrow(cookdata)
cookdata$ystar <- cookdata$DCBD - .5*cookdata$AGE
cookdata$y <- cookdata$ystar - mean(cookdata$ystar) + rnorm(n,sd=4) > 0

fit  <- cparlogit(y~DCBD+AGE,~LONGITUDE+LATITUDE,window=.5,
  distance="Latlong",data=cookdata,minp=0.001)
```

`cparlwr` *Conditionally Parametric LWR Estimation*

Description

Estimates a model of the form *y = XB(z) + u*, where *z* can include one or two variables. "Geographically weighted regression" is a special case in which *z = (latitude, longitude)* or some other measure of location.

Usage

```
cparlwr(form,nonpar,window=.25,bandwidth=0,kern="tcub",
  distance="Mahal",targetobs=NULL,data=NULL)
```

Arguments

`form`	Model formula
`nonpar`	List of either one or two variables for *z*. Formats: *cparlwr(y~xlist, nonpar=~z1, ...)* or *cparlwr(y~xlist, nonpar=~z1+z2, ...)*. Important: note the "~" before the first *z* variable.
`window`	Window size. Default: 0.25.
`bandwidth`	Bandwidth. Default: not used.
`kern`	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
`distance`	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *cparlwr* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
`targetobs`	If *targetobs = NULL*, uses the *maketarget* command to form targets. If *target="alldata"*, each observation is used as a target value for *x*. A set of target can also be supplied directly by listing the observation numbers of the target data points. The observation numbers can be identified using the *obs* variable produced by the *maketarget* command.

`data` A data frame containing the data. Default: use data in the current working directory

Details

The list of explanatory variables is specified in the base model formula while *Z* is specified using *nonpar*. The model formula does not have to include an intercept, making it suitable for repeat sales estimation as well as other models. *X* can include any number of explanatory variables, but *Z* must have at most two. *cparlwr* is equivalent to the *lwr* command when *Z* = *X* and the formula includes an intercept, with one exception: the explanatory variables are not centered on the target points so the intercept does not provide a direct estimate of y. This affects the intercept and its standard errors but not the coefficients on the explanatory variables. It also means that $\hat{y} = \hat{\alpha} + X\hat{\beta}$ rather than just $\hat{\alpha}$. The estimated coefficient matrix, *xcoef*, provides estimates of the slopes at z_0 , i.e., $B(z_0)$

The estimated value of *y* at a target value z_0 is the predicted value from a weighted least squares regression of *y* on *X* with weights given by *K*. When *Z* includes a single variable, *K* is a simple kernel weighting function: $K((z-z_0)/(sd(z)*h))$. When *Z* includes two variables (e.g., nonpar=~z1+z2), the method for specifying *K* depends on the *distance* option. Under either option, the *i*th row of the matrix *Z* = (z1, z2) is transformed such that $z_i = \sqrt{z_i * V * t(z_i)}$. Under the "Mahal" option, *V* is the inverse of cov(*Z*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*Z*)). After this transformation, the weights again reduce to the simple kernel weighting function $K((z - z_0)/(sd(z) * h))$.

The great circle formula is used to define *K* when *distance* = *"Latlong"*; in this case, the variable list for *nonpar* must be listed as *nonpar* = *~latitude+longitude* (or *~lo+la* or *~lat+long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude are listed does not matter and the function only looks for the first two letters to determine which variable is latitude and which is the longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance* = *"Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option.

h is specified by the *bandwidth* or *window* option. The function *cparlwrgrid* can be used to search for the value of *h* that minimizes the *cv* or *gcv* criterion.

Since each estimate is a linear function of all *n* values for *y*, the full set of estimates takes the form *yhat* = *LY*, where *L* is an *nxn* matrix. Loader (1999) suggests two measures of the number of degrees of freedom used in estimation: *df1* = *tr(L)* and *df2* = *tr(L'L)*. The diagonal elements of *tr(L)* are stored in the array *infl*. Since the degrees of freedom measures can differ substantially when *target="alldata"* rather than using a set of target points, it is a good idea to report final estimates using *target="alldata"* when possible.

Again following Loader (1999), the degrees of freedom correction used to estimate the error variance, *sig2*, is *df* = *2*df1 - df2*. Let *e* represent the vector of residuals, *y - yhat*. The estimated variance is $sig2 = \sum_i e_i^2/(n - df)$. The covariance matrix for $B(z_0)$ is

$$\hat{\sigma}^2(\sum_{i=1}^{n} X_i K(\phi_i) X_i^\top)^{-1}(\sum_{i=1}^{n} X_i (K(\phi_i))^2 X_i^\top)(\sum_{i=1}^{n} X_i K(\phi_i) X_i^\top)^{-1}.$$

Estimation can be very slow when *targetobs* = *"alldata"*. The *maketarget* command can be used to identify target points.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|)I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target` The target points for the original estimation of the function.

`ytarget` The predicted values of *y* at the target values *z*.

`xcoef.target` Estimated coefficients, *B(z)*, at the target values of *z*.

`xcoef.target.se` Standard errors for *B(z)* at the target values of *z*.

`yhat` Predicted values of *y* at the original data points.

`xcoef` Estimated coefficients, *B(z)*, at the original data points.

`xcoef.se` Standard errors for *B(z)* with *z* evaluated at all points in the data set.

`df1` *tr(L)*, a measure of the degrees of freedom used in estimation.

`df2` *tr(L'L)*, an alternative measure of the degrees of freedom used in estimation.

`sig2` Estimated residual variance, *sig2 = rss/(n-2*df1+df2)*.

`cv` Cross-validation measure. *cv = mean(((y-yhat)/(1-infl))^2)* , where *yhat* is the vector of predicted values for *y* and *infl* is the vector of diagonal terms for *L*.

`gcv` gcv = n*(n*sig2)/((n-nreg)^2), where *sig2* is the estimated residual variance and *nreg = 2*df1 - df2*.

`infl` A vector containing the diagonal elements of *L*.

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

McMillen, Daniel P., "One Hundred Fifty Years of Land Values in Chicago: A Nonparametric Approach," *Journal of Urban Economics* 40 (1996), 100-124.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

McMillen, Daniel P., "Employment Densities, Spatial Autocorrelation, and Subcenters in Large Metropolitan Areas," *Journal of Regional Science* 44 (2004), 225-243.

McMillen, Daniel P. and John F. McDonald, "A Nonparametric Analysis of Employment Density in a Polycentric City," *Journal of Regional Science* 37 (1997), 591-612.

McMillen, Daniel P. and Christian Redfearn, "Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions," *Journal of Regional Science* 50 (2010), 712-733.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics.* New York: Cambridge University Press, 1999.

See Also

cparlwrgrid

cubespline

fourier

lwr

lwrgrid

semip

Examples

```
data(cookdata)
par(ask=TRUE)
cookdata <- cookdata[cookdata$CHICAGO==1&!is.na(cookdata$LNFAR),]
fit1 <- cparlwr(LNFAR~DCBD,nonpar=~DCBD, window=.10,
  data=cookdata)
fit2 <- cparlwr(LNFAR~DCBD,nonpar=~LONGITUDE+LATITUDE,window=.10,
  distance="LATLONG",data=cookdata)
cookdata$yhat1 <- fit1$yhat
cookdata$yhat2 <- fit2$yhat
o <- order(cookdata$DCBD)
plot(cookdata$DCBD[o], cookdata$LNFAR[o],main="Log Floor Area Ratio",
  xlab="Distance from CBD",ylab="Log FAR")
lines(cookdata$DCBD[o], cookdata$yhat1[o], col="red")
plot(cookdata$DCBD[o], cookdata$LNFAR[o],main="Log Floor Area Ratio",
  xlab="Distance from CBD",ylab="Log FAR")
points(cookdata$DCBD[o], cookdata$yhat2[o], col="red")
```

`cparlwrgrid` *Conditionally parametric LWR regression bandwidth or window selection*

Description

Finds the value of a user-provided array of window or bandwidth values that provides the lowest *cv* or *gcv* for a *CPAR* model. Calls *cparlwr* and returns its full output for the chosen value of *h*.

Usage

```
cparlwrgrid(form,nonpar,window=0,bandwidth=0,kern="tcub",method="gcv",
  print=TRUE,distance="Mahal",targetobs=NULL,data=NULL)
```

Arguments

form	Model formula
nonpar	List of either one or two variables for *z*. Formats: *cparlwr(y~xlist, nonpar=~z1, ...)* or *cparlwr(y~xlist, nonpar=~z1+z2, ...)*. Important: note the "~" before the first *z* variable.
window	Window size. Default: not used.
bandwidth	Bandwidth. Default: not used.
kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
method	Specifies "gcv" or "cv" criterion function. Default: method="gcv".
print	If TRUE, prints *gcv* or *cv* values for each value of the window or bandwidth.
distance	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *cparlwr* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
targetobs	If *targetobs = NULL*, uses the *maketarget* command to form targets. If *target="alldata"*, each observation is used as a target value for *x*. A set of target can also be supplied directly by listing the observation numbers of the target data points. The observation numbers can be identified using the *obs* variable produced by the *maketarget* command.
data	A data frame containing the data. Default: use data in the current working directory

Value

target	The target points for the original estimation of the function.
ytarget	The predicted values of *y* at the target values *z*.
xcoef.target	Estimated coefficients, *B(z)*, at the target values of *z*.
xcoef.target.se	Standard errors for *B(z)* at the target values of *z*.
yhat	Predicted values of *y* at the original data points.
xcoef	Estimated coefficients, *B(z)*, at the original data points.
xcoef.se	Standard errors for *B(z)* with *z* evaluated at all points in the data set.
df1	*tr(L)*, a measure of the degrees of freedom used in estimation.
df2	*tr(L'L)*, an alternative measure of the degrees of freedom used in estimation.

sig2	Estimated residual variance, *sig2 = rss/(n-2*df1+df2)*.
cv	Cross-validation measure. *cv = mean(((y-yhat)/(1-infl))^2)* , where *yhat* is the vector of predicted values for *y* and *infl* is the vector of diagonal terms for *L*.
gcv	gcv = n*(n*sig2)/((n-nreg)^2), where *sig2* is the estimated residual variance and *nreg = 2*df1 - df2*.
infl	A vector containing the diagonal elements of *L*.

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

McMillen, Daniel P., "One Hundred Fifty Years of Land Values in Chicago: A Nonparametric Approach," *Journal of Urban Economics* 40 (1996), 100-124.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

McMillen, Daniel P., "Employment Densities, Spatial Autocorrelation, and Subcenters in Large Metropolitan Areas," *Journal of Regional Science* 44 (2004), 225-243.

McMillen, Daniel P. and John F. McDonald, "A Nonparametric Analysis of Employment Density in a Polycentric City," *Journal of Regional Science* 37 (1997), 591-612.

McMillen, Daniel P. and Christian Redfearn, "Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions," *Journal of Regional Science* 50 (2010), 712-733.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics.* New York: Cambridge University Press, 1999.

See Also

cparlwr

Examples

```
par(ask=TRUE)
n = 1000
z1 <- runif(n,0,2*pi)
z1 <- sort(z1)
z2 <- runif(n,0,2*pi)
o1 <- order(z1)
o2 <- order(z2)
ybase1 <-  z1 - .1*(z1^2) + sin(z1) - cos(z1) - .5*sin(2*z1) + .5*cos(2*z1)
ybase2 <- -z2 + .1*(z2^2) - sin(z2) + cos(z2) + .5*sin(2*z2) - .5*cos(2*z2)
ybase <- ybase1+ybase2
sig = sd(ybase)/2
y <- ybase + rnorm(n,0,sig)
summary(lm(y~ybase))

# Single variable estimation
```

```
fit1 <- cparlwrgrid(y~z1,nonpar=~z1,window=seq(.10,.40,.10))
c(fit1$df1,fit1$df2,2*fit1$df1-fit1$df2)
plot(z1[o1],ybase1[o1],type="l",ylim=c(min(ybase1,fit1$yhat),max(ybase1,fit1$yhat)),
  xlab="z1",ylab="y")
# Make predicted and actual values have the same means
fit1$yhat <- fit1$yhat - mean(fit1$yhat) + mean(ybase1)
lines(z1[o1],fit1$yhat[o1], col="red")
legend("topright", c("Base", "LWR"), col=c("black","red"),lwd=1)
fit2 <- cparlwrgrid(y~z2,nonpar=~z2,window=seq(.10,.40,.10))
fit2$yhat <- fit2$yhat - mean(fit2$yhat) + mean(ybase2)
c(fit2$df1,fit2$df2,2*fit2$df1-fit2$df2)
plot(z2[o2],ybase2[o2],type="l",ylim=c(min(ybase2,fit2$yhat),max(ybase2,fit2$yhat)),
    xlab="z1",ylab="y")
lines(z2[o2],fit2$yhat[o2], col="red")
legend("topright", c("Base", "LWR"), col=c("black","red"),lwd=1)

#both variables
fit3 <- cparlwrgrid(y~z1+z2,nonpar=~z1+z2,window=seq(.10,.20,.05))
yhat1 <- fit3$yhat - mean(fit3$yhat) + mean(ybase1)
plot(z1[o1],yhat1[o1], xlab="z1",ylab="y")
lines(z1[o1],ybase1[o1],col="red")
yhat2 <- fit3$yhat - mean(fit3$yhat) + mean(ybase2)
plot(z2[o2],yhat2[o2], xlab="z2",ylab="y")
lines(z2[o2],ybase2[o2],col="red")
```

`cparmlogit` *Conditionally parametric logit for two or more choices*

Description

Estimates a multinomial logit model with two or more choices by maximizing a locally weighted likelihood function – the logit equivalent of cparlwr

Usage

```
cparmlogit(form,nonpar,window=.25,bandwidth=0,kern="tcub",
distance="Mahal",target=NULL,data=NULL)
```

Arguments

`form`	Model formula
`nonpar`	List of either one or two variables for *z*. Formats: *cparmlogit(y~xlist, nonpar=~z1, ...)* or *cparmlogit(y~xlist, nonpar=~z1+z2, ...)*. Important: note the "~" before the first *z* variable.
`window`	Window size. Default: 0.25.
`bandwidth`	Bandwidth. Default: not used.

kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
distance	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *cparmlogit* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
target	If *target* = *NULL*, uses the *maketarget* command to form targets using the values specified for *window*, *bandwidth*, and *kern*. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
data	A data frame containing the data. Default: use data in the current working directory

Details

The list of explanatory variables is specified in the base model formula while *Z* is specified using *nonpar*. *X* can include any number of explanatory variables, but *Z* must have at most two.

The model is estimated by maximizing the following weighted log-likelihood function at each target point:

$$\sum_{i=1}^{n}\sum_{j=1}^{K} w_i I(y_i = j) log(P(X_i\beta_j))$$

where y is the discrete dependent variable with K+1 choices, X is the set of explanatory variables, and $P(X_i\beta_j) = \frac{exp(X_i\beta_j)}{\sum_j exp(X_i\beta_j)}$. For the base value, y=0, the coefficients are normalized to $\beta_0 = 0$.

When *Z* includes a single variable, w_i is a simple kernel weighting function: $w_i = K((z_i - z_0)/(sd(z) * h))$. When *Z* includes two variables (e.g., nonpar=~z1+z2), the method for specifying *w* depends on the *distance* option. Under either option, the *i*th row of the matrix *Z* = (z1, z2) is transformed such that $z_i = \sqrt{z_i * V * t(z_i)}$. Under the "Mahal" option, *V* is the inverse of cov(*Z*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*Z*)). After this transformation, the weights again reduce to the simple kernel weighting function $K((z_i - z_0)/(sd(z) * h))$. *h* is specified by the *bandwidth* or *window* option.

The great circle formula is used to constuct the distances used to form the weights when *distance* = *"Latlong"*; in this case, the variable list for *nonpar* must be listed as *nonpar* = *~latitude+longitude* (or *~lo+la* or *~lat+long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude are listed does not matter and the function only looks for the first two letters to determine which variable is latitude and which is the longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance* = *"Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option.

Following White (1982), the covariance matrix for a quasi-maximum likelihood model is $A^{-1}BA^{-1}$, where

$$A = \sum_{i=1}^{n} w_i \frac{\partial^2 LnL_i}{\partial\beta\partial\beta'}$$

$$B = \sum_{i=1}^{n} w_i^2 \frac{\partial LnL_i}{\partial\beta} \frac{\partial LnL_i}{\partial\beta'}$$

The covariance matrix is calculated at each target point and the implied standard errors are then interpolated to each data point. Estimation can be very slow when *target = "alldata"*. The *maketarget* command can be used to identify target points.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|)I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target`	The target points for the original estimation of the function.
`xcoef.target`	Estimated coefficients, *B(z)*, at the target values of *z*.
`xcoef.target.se`	Standard errors for *B(z)* at the target values of *z*.
`xcoef`	Estimated coefficients, *B(z)*, at the original data points.
`xcoef.se`	Standard errors for *B(z)* with *z* evaluated at all points in the data set.
`pmat`	The n x K+1 matrix of estimated probabilities.
`lnl`	The log-likelihood value.

References

Fan, Jianqing, Nancy E. Heckman, and M.P. Wand, "Local Polynomial Kernel Regression for Generalized Linear Models and Quasi-Likelihood Functions," *Journal of the American Statistical Association* 90 (1995), 141-150.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

McMillen, Daniel P. and John F. McDonald, "Locally Weighted Maximum Likelihood Estimation: Monte Carlo Evidence and an Application," in Luc Anselin, Raymond J.G.M. Florax, and Sergio J. Rey, eds., *Advances in Spatial Econometrics*, Springer-Verlag, New York (2004), 225-239.

Tibshirani, Robert and Trevor Hastie, "Local Likelihood Estimation," *Journal of the American Statistical Association* 82 (1987), 559-568.

See Also

cparlogit

cparprobit

gmmlogit

gmmprobit

splogit

spprobit

spprobitml

Examples

```
library(mlogit)
set.seed(5647)
n = 1000
x <- runif(n,0,pi*sqrt(12))
o <- order(x)
x <- x[o]
form <- yvar~x
nonpar <- ~x

# 2 choices
ybase <- x + rlogis(n)
yvar <- ybase>.5*pi*sqrt(12)
table(yvar)
fit <- glm(yvar~x,family=binomial(link="logit"))
summary(fit)
p <- fitted(fit)
fit1 <- cparmlogit(yvar~x,nonpar=~x,window=.5,kern="tcub")
fit1$lnl
colMeans(fit1$xcoef)
colMeans(fit1$xcoef.se)
cor(p,fit1$pmat)
plot(x,p,xlab="x",ylab="Prob(y=1)",type="l")
lines(x,fit1$pmat[,2],col="red")
legend("topleft",c("Standard Logit","CPAR"),col=c("black","red"),lwd=1)

## Not run:
par(ask=TRUE)
# 3 choices
ybase1 <- -.5*pi*sqrt(12) + x + rlogis(n)
ybase2 <-  -.5*pi*sqrt(12)/2 + x/2 + rlogis(n)
yvar <- ifelse(ybase1>ybase2,1,2)
yvar <- ifelse(ybase1<0&ybase2<0,0,yvar)
table(yvar)
mdata <- data.frame(yvar,x)
fit <- mlogit(yvar~0 | x, data=mdata, shape="wide")
summary(fit)
fit1 <- cparmlogit(yvar~x,nonpar=~x,window=.5,kern="tcub")
fit1$lnl
```

```
colMeans(fit1$xcoef)
colMeans(fit1$xcoef.se)
cor(fit$probabilities,fit1$pmat)
plot(x,fit$probabilities[,1],xlab="x",ylab="Prob(y=1)",type="l",main="Prob(y=0)")
lines(x,fit1$pmat[,1],col="red")
legend("topright",c("Standard Logit","CPAR"),col=c("black","red"),lwd=1)
plot(x,fit$probabilities[,2],xlab="x",ylab="Prob(y=1)",type="l",main="Prob(y=1)")
lines(x,fit1$pmat[,2],col="red")
legend("topleft",c("Standard Logit","CPAR"),col=c("black","red"),lwd=1)
plot(x,fit$probabilities[,3],xlab="x",ylab="Prob(y=1)",type="l",main="Prob(y=2)")
lines(x,fit1$pmat[,3],col="red")
legend("topleft",c("Standard Logit","CPAR"),col=c("black","red"),lwd=1)

# 2 choices, quadratic
x2 <- x^2
ybase <- x - .1*(x^2) + rlogis(n)
yvar <- ybase>median(ybase)
table(yvar)
fit <- glm(yvar~x+x2,family=binomial(link="logit"))
summary(fit)
p <- fitted(fit)
fit1 <- cparmlogit(yvar~x,nonpar=~x,window=.25,kern="tcub")
fit1$lnl
colMeans(fit1$xcoef)
colMeans(fit1$xcoef.se)
cor(p,fit1$pmat)
plot(x,p,xlab="x",ylab="Prob(y=1)",type="l")
lines(x,fit1$pmat[,2],col="red")
legend("topleft",c("Standard Logit","CPAR"),col=c("black","red"),lwd=1)

## End(Not run)
```

`cparprobit` *Conditionally Parametric probit for two choices*

Description

Estimates a probit model with two choices by maximizing a locally weighted likelihood function – the probit equivalent of cparlwr

Usage

```
cparprobit(form,nonpar,window=.25,bandwidth=0,kern="tcub",
distance="Mahal",target=NULL,data=NULL,minp=NULL)
```

Arguments

`form`	Model formula

nonpar	List of either one or two variables for *z*. Formats: *cparprobit(y~xlist, nonpar=~z1, ...)* or *cparprobit(y~xlist, nonpar=~z1+z2, ...)*. Important: note the "~" before the first *z* variable.
window	Window size. Default: 0.25.
bandwidth	Bandwidth. Default: not used.
kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
distance	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *cparprobit* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
target	If *target* = *NULL*, uses the *maketarget* command to form targets using the values specified for *window*, *bandwidth*, and *kern*. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
data	A data frame containing the data. Default: use data in the current working directory
minp	Specifies a limit for the estimated probability. Any estimated probability lower than *minp* will be set to *minp* and any probability higher than 1-*minp* will be set to 1-*minp*. By default, the estimated probabilities are bounded by 0 and 1.

Details

The list of explanatory variables is specified in the base model formula while *Z* is specified using *nonpar*. *X* can include any number of explanatory variables, but *Z* must have at most two.

The model is estimated by maximizing the following weighted log-likelihood function at each target point:

$$\sum_{i=1}^{n} w_i\{y_i log(\Phi(X_i\beta)) + (1-y_i) log(1-\Phi(X_i\beta))\}$$

where y is the discrete dependent variable and X is the set of explanatory variables.

When *Z* includes a single variable, w_i is a simple kernel weighting function: $w_i = K((z_i - z_0)/(sd(z) * h))$. When *Z* includes two variables (e.g., nonpar=~z1+z2), the method for specifying *w* depends on the *distance* option. Under either option, the *i*th row of the matrix *Z* = (z1, z2) is transformed such that $z_i = \sqrt{z_i * V * t(z_i)}$. Under the "Mahal" option, *V* is the inverse of cov(*Z*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*Z*)). After this transformation, the weights again reduce to the simple kernel weighting function $K((z_i - z_0)/(sd(z) * h))$. *h* is specified by the *bandwidth* or *window* option.

The great circle formula is used to construct the distances used to form the weights when *distance* = *"Latlong"*; in this case, the variable list for *nonpar* must be listed as *nonpar* = *~latitude+longitude* (or *~lo+la* or *~lat+long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude are listed does not matter and the function only looks for the first

two letters to determine which variable is latitude and which is longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance = "Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option.

Following White (1982), the covariance matrix for a quasi-maximum likelihood model is $A^{-1}BA^{-1}$, where

$$A = \sum_{i=1}^{n} w_i \frac{\partial^2 LnL_i}{\partial\beta\partial\beta'}$$

$$B = \sum_{i=1}^{n} w_i^2 \frac{\partial LnL_i}{\partial\beta} \frac{\partial LnL_i}{\partial\beta'}$$

For the probit model,

$$A = \sum_{i=1}^{n} w_i P_i(1-P_i) X_i X_i'$$

$$B = \sum_{i=1}^{n} w_i^2 (y_i - P_i)^2 X_i X_i'$$

The covariance matrix is calculated at all target points and the implied standard errors are then interpolated to each data point.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\|<1)$
Triangular	"tria"	$(1-\|z\|)I(\|z\|<1)$
Epanechnikov	"epan"	$\frac{3}{4}(1-z^2)*I(\|z\|<1)$
Bi-Square	"bisq"	$\frac{15}{16}(1-z^2)^2*I(\|z\|<1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1-\|z\|^3)^3*I(\|z\|<1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1-z^2)^3*I(\|z\|<1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target`	The target points for the original estimation of the function.
`xcoef.target`	Estimated coefficients, *B(z)*, at the target values of *z*.
`xcoef.target.se`	Standard errors for *B(z)* at the target values of *z*.
`xcoef`	Estimated coefficients, *B(z)*, at the original data points.
`xcoef.se`	Standard errors for *B(z)* with *z* evaluated at all points in the data set.
`p`	The estimated probabilities.
`lnl`	The log-likelihood value.

References

Fan, Jianqing, Nancy E. Heckman, and M.P. Wand, "Local Polynomial Kernel Regression for Generalized Linear Models and Quasi-Likelihood Functions," *Journal of the American Statistical Association* 90 (1995), 141-150.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

McMillen, Daniel P. and John F. McDonald, "Locally Weighted Maximum Likelihood Estimation: Monte Carlo Evidence and an Application," in Luc Anselin, Raymond J.G.M. Florax, and Sergio J. Rey, eds., *Advances in Spatial Econometrics*, Springer-Verlag, New York (2004), 225-239.

Tibshirani, Robert and Trevor Hastie, "Local Likelihood Estimation," *Journal of the American Statistical Association* 82 (1987), 559-568.

See Also

cparlogit

cparmlogit

gmmlogit

gmmprobit

splogit

spprobit

spprobitml

Examples

```
set.seed(5647)
data(cookdata)
cookdata <- cookdata[!is.na(cookdata$AGE),]
n = nrow(cookdata)
cookdata$ystar <- cookdata$DCBD - .5*cookdata$AGE
cookdata$y <- cookdata$ystar - mean(cookdata$ystar) + rnorm(n,sd=4) > 0

tvect <- maketarget(~LONGITUDE+LATITUDE,window=.5,data=cookdata)$target
fit  <- cparprobit(y~DCBD+AGE,~LONGITUDE+LATITUDE,window=.5,
  target=tvect,distance="Latlong",data=cookdata,minp=0.001)
```

---

cubespline *Smooth cubic spline estimation*

Description

Estimates a smooth cubic spline for a model of the form y = f(z) + XB +u. The function divides the range of z into knots+1 equal intervals. The regression is $y = cons + \lambda_1(z - z_0) + \lambda_2(z - z_0)^2 + \lambda_3(z - z_0)^3 + \sum_{k=1}^{K} \gamma_k(z - z_k)^3 D_k + X\beta + u$ where $z_0 = min(z), z_1 ... z_K$ are the knots, and $D_k = 1$ if $z \geq z_k$. Estimation can be carried out for a fixed value of K or for a range of K. In the latter case, the function indicates the value of K that produces the lowest value of one of the following criteria: the AIC, the Schwarz information criterion, or the gcv.

Usage

```
cubespline(form,knots=1,mink=1,maxk=1,crit="gcv",data=NULL)
```

Arguments

form	Model formula. The spline is used with the first explanatory variable.
knots	If knots is specified, fits a cubic spline with K = knots. Default is knots=1, mink = 1, and maxk = 1, which implies a cubic spline with a single knot.
mink	The lower bound to search for the value of K that minimizes crit. mink can take any value greater than zero. The default is mink = 1
maxk	The upper bound to search for the value of K that minimizes crit. maxk must be great than or equal to mink. The default is maxk = 1.
crit	The selection criterion. Must be in quotes. The default is the generalized cross-validation criterion, or "gcv". Options include the Akaike information criterion, "aic", and the Schwarz criterion, "sc". Let nreg be the number of explanatory variables in the regression and sig2 the estimated variance. The formulas for the available crit options are gcv = n*(n*sig2)/((n-nreg)^2) aic = log(sig2) + 2* nreg /n sc = log(sig2) + log(n)*nreg /n
data	A data frame containing the data. Default: use data in the current working directory

Value

yhat	The predicted values of the dependent variable at the original data points
rss	The residual sum of squares
sig2	The estimated error variance
aic	The value for AIC
sc	The value for sc
gcv	The value for gcv
coef	The estimated coefficient vector, B
splinehat	The predicted values for z alone, normalized to have the same mean as the dependent variable. If no X variables are included in the regression, splinehat = yhat.
knots	The vector of knots

References

McMillen, Daniel P., "Testing for Monocentricity," in Richard J. Arnott and Daniel P. McMillen, eds., *A Companion to Urban Economics*, Blackwell, Malden MA (2006), 128-140.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

Suits, Daniel B., Andrew Mason, and Louis Chan, "Spline Functions Fitted by Standard Regression Methods," *Review of Economics and Statistics* 60 (1978), 132-139.

See Also

cparlwr

fourier

lwr

lwrgrid

semip

Examples

```
data(cookdata)
fardata <- cookdata[!is.na(cookdata$LNFAR),]
par(ask=TRUE)

# single variable
o <- order(fardata$DCBD)
fit1 <- cubespline(LNFAR~DCBD, mink=1, maxk=10,data=fardata)
c(fit1$rss, fit1$sig2, fit1$aic, fit1$sc, fit1$gcv, fit1$knots)
plot(fardata$DCBD[o], fardata$LNFAR[o], xlab="Distance from CBD", ylab="Log FAR")
lines(fardata$DCBD[o], fit1$splinehat[o], col="red")

# multiple explanatory variables
fit2 <- cubespline(fardata$LNFAR~fardata$DCBD+fardata$AGE,  mink=1, maxk=10)
c(fit2$rss, fit2$sig2, fit2$aic, fit2$sc, fit2$gcv, fit2$knots)
plot(fardata$DCBD[o], fardata$LNFAR[o], xlab="Distance from CBD", ylab="Log FAR")
lines(fardata$DCBD[o], fit2$splinehat[o], col="red")

# pre-specified number of knots
fit3 <- cubespline(LNFAR~DCBD+AGE,  knots=4, data=fardata)
```

---

dfldens *Counterfactual Kernel Density Functions*

Description

Uses the DiNardo, Fortin, and Lemieux approach to re-weight kernel density functions based on values of an explanatory variable from an earlier period.

Usage

```
dfldens(y,lgtform,window=0,bandwidth=0,kern="tcub",probit=FALSE,
  graph=TRUE,yname="y",alldata=FALSE,data=NULL)
```

Arguments

y	The dependent variable for which the counterfactual density is estimated. The data frame must be specified if it has not been attached, e.g., *y=mydata$depvar.*
lgtform	The formula for the logit or probit model for the time variable. The dependent variable should be a 0-1 variable with 1's representing the later time period. Example: *lgtform=timevar~x1+x2.*
window	The window size for the kernel density function. Default: not used.
bandwidth	The bandwidth. Default: bandwidth = (.9*(quantile(y1,.75)-quantile(y1,.25))/1.34)*(n1^(-.20)), specified by setting *bandwidth* = 0 and *window* = 0.
kern	Kernel weighting function. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
probit	If *TRUE*, a probit model is used for the time variable rather than logit. Default: *probit = FALSE*.
graph	If *TRUE*, produces a graph showing the density function for time 1 and the counterfactual density. Default: *graph=TRUE*.
yname	The name to be used for the variable whose density functions are drawn when *graph=T*. Default: *yname = "y"*.
alldata	If *TRUE*, the density functions are calculated using each observation in turn as a target value. When *alldata=F*, densities are calculated at a set of points chosen by the *locfit* program using an adaptive decision tree approach, and the *smooth12* command is used to interpolate to the full set of observations.
data	A data frame with the variables for the logit or probit model specified by *lgtform*. Note: the data frame for *y* must be specified even if it is part of *data*.

Details

The *dfldens* command first calculates kernel density estimates for *y* in time period *timevar* = 1. The density estimate at target point *y* is $f(y_1) = (1/(hn_1)) \sum_i K((y_{1i} - y_1)/h)$. The following kernel weighting functions are available:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2} I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|) I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5} e^{-z^2/2}$

By default, *dfldens* uses a tri-cube kernel with a fixed bandwidth of h = (.9*(quantile(y1,.75)-quantile(y1,.25))/1.34)*(n1^(-.20)). The results are stored in *dtarget1* and *dhat1*.

The counterfactual density is an estimate of the density function for *y* in time 1 if the explanatory variables listed in *lgtform* were equal to their time 0 values. DiNardo, Fortin, and Lemieux (1996) show that the the following re-weighting of $f(y_1)$ is an estimate of the counterfactual density:

$(1/(hn_1))\sum_i \tau_i K((y_{1i}-y_1)/h)$. The weights are given by $tau_i = (P(x_i)/(1-P(x_i)))/(p/(1-p))$, where $p = n_0/(n_0+n_1))$ and $P(x_i))$ is the estimated probability that *timevar = 0* from the estimated logit or probit regression of *timevar* on *X*.

If *X* includes a single variable *x*, the counterfactual density shows how the $f(y_1)$ would change if $x = x_0$ rather than x_1. Alternatively, *X* can include multiple variables, in which case the counterfactual density shows how the $f(y_1)$ would change if all of the variables in *X* were equal to their *timevar = 0* values.

Value

`target`	The vector of target values for *y* for the density functions.
`dtarget1`	The vector of densities in period 1 at the target values of *y*.
`dtarget10`	The counterfactual densities in period 1 at the target values of *y*.
`dhat1`	The vector of densities in period 1 at the actual values of *y*.
`dhat10`	The counterfactual densities in period 1 at the actual values of *y*.

References

DiNardo, J., N. Fortin, and T. Lemieux, "Labor Market Institutions and the Distribution of Wages, 1973-1992: A Semi-Parametric Approach," *Econometrica* 64 (1996), 1001-1044.

Leibbrandt, Murray, James A. Levinsohn, and Justin McCrary, "Incomes in South Africa after the Fall of Apartheid," *Journal of Globalization and Development* 1 (2010).

See Also

`qregsim2`

Examples

```
data(matchdata)
matchdata$year05 <- matchdata$year==2005
fit <- dfldens(matchdata$lnprice, year05~lnland+lnbldg, window=.2,
  yname = "Log of Sale Price", data=matchdata)
matchdata$age <- matchdata$year - matchdata$yrbuilt
fit <- dfldens(matchdata$lnprice, year05~age, window=.2,
  yname="Log of Sale Price", data=matchdata)
```

---

dupage99 *DuPage County assessment ratio data set*

Description

A random draw of 2000 assessment ratios from DuPage County, IL in 1999. Sales took place in 1999; the assessments were in place in 1998. Statutory assessment rates in DuPage County were 0.33.

Usage

```
data(dupage99)
```

Format

A data frame with 2000 observations on the following 2 variables.

`av` the first column. The assessed value.

`price` the second column. The sales price.

Source

Daniel McMillen. Data were provided originally by the Illinois Department of Revenue.

`fourier` *Fourier expansion smoothing*

Description

Estimates a model of the form y=f(z)+XB+u using a fourier expansion for z. The variable z is first transformed to $z = 2\pi\frac{(z-min(z))}{max(z)-min(z))}$. The fourier model is $y = \alpha_1 z + \alpha_2 z^2 + \sum(\lambda_q sin(qz) + \delta_q cos(qz)) + X\beta + u$. Estimation can be carried out for a fixed value of Q or for a range of Q. In the latter case, the function indicates the value of Q that produces the lowest value of one of the following criteria: the AIC, the Schwarz information criterion, or the gcv.

Usage

```
fourier(form,q=1,minq=0,maxq=0,crit="gcv",data=NULL)
```

Arguments

`form`	Model formula. The expansion is applied to the first explanatory variable.
`q`	If q is specified and minq=maxq, fits a fourier expansion with Q set to q. Default is q=1, which implies a model with z, z^2, sin(z), cos(z) and X as explanatory variables.
`minq`	The lower bound to search for the value of Q that minimizes crit. minq can take any value greater than zero. Default: not used.
`maxq`	The upper bound to search for the value of Q that minimizes crit. maxq must be great than or equal to minq. Default: not used.
`crit`	The selection criterion. Must be in quotes. The default is the generalized cross-validation criterion, or "gcv". Options include the Akaike information criterion, "aic", and the Schwarz criterion, "sc". Let nreg be the number of explanatory variables in the regression and sig2 the estimated variance. The formulas for the available crit options are gcv = n*(n*sig2)/((n-nreg)^2) aic = log(sig2) + 2* nreg /n sc = log(sig2) + log(n)*nreg /n

data	A data frame containing the data. Default: use data in the current working directory

Value

yhat	The predicted values of the dependent variable at the original data points
rss	The residual sum of squares
sig2	The estimated error variance
aic	The value for AIC
sc	The value for sc
gcv	The value for gcv
coef	The estimated coefficient vector, B
fourierhat	The predicted values for z alone, normalized to have the same mean as the dependent variable. If no X variables are included in the regression, fourierhat = yhat.
q	The value of Q used in the final estimated model.

References

Gallant, Ronald, "On the Bias in Flexible functional Forms and an Essentially Unbiased Form: The Fourier Flexible Form," *Journal of Econometrics* 15 (1981), 211-245.

Gallant, Ronald, "Unbiased Determination of Production Technologies," *Journal of Econometrics* 20 (1982), 285-323.

McMillen, Daniel P. and Jonathan Domborw, "A Flexible Fourier Approach to Repeat Sales Price Indexes," *Real Estate Economics* 29 (2001), 207-225.

McMillen, Daniel P., "Neighborhood Price Indexes in Chicago: A Fourier Repeat Sales Approach," *Journal of Economic Geography* 3 (2003), 57-73.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

See Also

cparlwr

cubespline

lwr

lwrgrid

semip

Examples

```
set.seed(23849103)
n = 1000
x <- runif(n,0,2*pi)
x <- sort(x)
ybase <- x - .1*(x^2) + sin(x) - cos(x) -.5*sin(2*x) + .5*cos(2*x)
```

```
sig = sd(ybase)/2
y <- ybase + rnorm(n,0,sig)

par(ask=TRUE)
plot(x,y)
lines(x,ybase,col="red")

fit <- fourier(y~x,minq=1,maxq=10)
plot(x,ybase,type="l",xlab="x",ylab="y")
lines(x,fit$yhat,col="red")
legend("topright",c("Base","Fourier"),col=c("black","red"),lwd=1)
```

---

geodensity *Kernel density functions for geo-coded data*

Description

Calculates kernel density functions for geo-coded data based on straight-line distances between observations

Usage

```
geodensity(longvar,latvar,window=.25,kern="tcub",alldata=FALSE)
```

Arguments

longvar	Longitude variable, in degrees.
latvar	Latitude variable, in degrees.
window	Window size. Default: 0.25.
kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", and "trwt".
alldata	If *alldata=T*, each observation is used as a target value for *x*. When *alldata=F*, the function is estimated at a set of points chosen by the *locfit* program using an adaptive decision tree approach, and the *smooth12* command is used to interpolate to the full set of observations. Specifying *alldata=T* can lead to long estimation times.

Details

The *geodistance* function is used to calculate straight-line distances between all observations and each target point. The vector of distances for a given target value is *d*. The window is determined by finding $dmax = quantile(d, window)$. The estimated density at the target point is simply:

$$f = \frac{1}{dmax * n} \sum_i K(\frac{d_i}{dmax})$$

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|)I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$

The gaussian kernel is not available.

If *alldata=T*, each data point in turn is used as a target point. If *alldata=F*, *locfit* is used to find a set of target points, and the *smooth12* command is used to interpolate to the full set of observations. The matrix of target coordinates is stored in *target*, and the estimated densities at the target points are stored in *dens.target*. If *alldata=T*, *target* contains the full set of values for longitude and latitude, and *dens.target* = *denshat*.

Value

`target`	The matrix of target values. Dimensions = ntx2, where nt is the number of target points. First column = longitude, second column = latitude.
`dens.target`	The estimated densities at the target coordinates.
`denshat`	The estimated densities at the original data points.

See Also

geodistance

geogravity

Examples

```
## Not run:
library(spdep)
library(RColorBrewer)
cook <- readShapePoly(system.file("maps/CookCensusTracts.shp",package="McSpatial"))
# measure distance to Chicago city center
lmat <- coordinates(cook)
cook$longitude <- lmat[,1]
cook$latitude  <- lmat[,2]
fit <- geodensity(cook$longitude, cook$latitude)
cook$denshat <- fit$denshat
brks <- seq(min(cook$denshat,na.rm=TRUE),max(cook$denshat,na.rm=TRUE),length=9)
spplot(cook,"denshat",at=brks,col.regions=rev(brewer.pal(9,"RdBu")),
  main="Census Tract Densities")

## End(Not run)
```

`geodistance` *Calculates distances using the great circle formula*

Description

geodistance calculates distances in miles between a set of observations and a location. Distances are calculated in miles using the great circle formula. Geographic coordinates must be expressed in latitudes and longitudes.

Usage

```
geodistance(longvar, latvar, lotarget, latarget, dcoor = FALSE)
```

Arguments

`longvar`	Longitude variable, in degrees.
`latvar`	Latitude variable, in degrees.
`lotarget`	Target longitude.
`latarget`	Target latitude.
`dcoor`	If dcoor = T, also calculates the distance east *(deast)* and north *(dnorth)* of the target point.

Value

`dist`	A vector with the distance in miles between each data point and the target point.
`dnorth`	A vector with the number of miles north of the target point for each data point. *dnorth* < 0 for observations that are south of the target point.
`deast`	A vector with the number of miles east of the target point for each data point. *deast* < 0 for observations that are west of the target point.

See Also

geodensity

geoshape

Examples

```
data(cookdata)
dcbd <- geodistance(cookdata$LONGITUDE, cookdata$LATITUDE, -87.627800, 41.881998)$dist
```

`geogravity` *Gravity matrix and gravity variable calculations*

Description

Calculates a variable showing the average value for each observation of a gravity measure of the spatial interaction between the observation and the other points in a data set.

Usage

```
geogravity(x,longvar,latvar,alpha=1,maxd=NULL,alldata=FALSE,
  window=.10,outmatrix=FALSE)
```

Arguments

`x`	The variable of interest, e.g., population or employment.
`longvar`	Longitude variable, in degrees.
`latvar`	Latitude variable, in degrees.
`alpha`	The α parameter for the distance variables. Default: *alpha* = 1.
`maxd`	Maximum distance, beyond which observations get zero weight. Default: all observations are included in the calculations.
`alldata`	If *FALSE*, interpolates between target points rather than making the calculations at every observation. Default: *alldata=FALSE*.
`window`	Window size used to determine a set of target points when *alldata=FALSE*. Default: *window* = .10.
`outmatrix`	If *TRUE* and *alldata==TRUE*, stores the full matrix of gravity values. Default: *outmatrix=FALSE*.

Details

The gravity measure of the spatial interaction between two points *i* and *j* is $g_{ij} = P_i P_j / d_{ij}^{\alpha}$ if $d_{ij} \leq maxd$ and $g_{ij} = 0$ if $d_{ij} > maxd$, where *P* is a variable such as population or employment that measures the importance of the observation, d_{ij} is the straight-line distance between observations *i* and *j*, and α is a parameter. The variable *P* is provided by the argument *x*. The full $n \times n$ matrix of values is stored in *dmat* if *alldata=TRUE* and *outmatrix=TRUE*, with the diagonal elements set to zero.

The "gravity variable" that is stored in *gravity* is the average value for each observation:

$$g_i = \frac{1}{n-1} \sum_{j \neq i} g_{ij}$$

By default, the *locfit* program is used to find a set of target values for calculating g_{ij}. The full set of observations is used for the *j* index, but a smaller set of observations is used for the target values, *i*. The abbreviated set of g_i values is then interpolated to the full set of data points using the *smooth12* command. The vector of target observation numbers is stored in *targetobs*.

Value

targetobs	The vector of target observation numbers.
gtarget	The gravity variable at the target points.
gravity	The gravity variable for the full data set.
dmat	The full nxn matrix of results, if *outmatrix=TRUE* and *alldata=TRUE*. The diagonal elements are set to zero.

See Also

geodistance

geodensity

Examples

```
library(spdep)
cook <- readShapePoly(system.file("maps/CookCensusTracts.shp",package="McSpatial"))
cook <- cook[cook$POPULATION>0&cook$AREA>0,]
cook$lndens <- log(cook$POPULATION/cook$AREA)
lmat <- coordinates(cook)
longitude <- lmat[,1]
latitude  <- lmat[,2]
fit <- geogravity(cook$lndens,longitude,latitude)
cook$gravity <- fit$gravity
```

---

geoshape *Calculates distances between a shape file and a set of points.*

Description

Calculates distances in miles between a shape file and a set of geographic coordinates.

Usage

```
geoshape(longvar,latvar,linefile=NULL,pointfile=NULL,coormatrix=NULL)
```

Arguments

longvar	Longitude variable, in degrees.
latvar	Latitude variable, in degrees.
linefile	A shape file with lines. In this case, *geoshape* calculates the distance between each observation and the nearest line in the shape file. Default: *linefile=NULL*.
pointfile	A shape file with points. In this case, *geoshape* calculates the distance between each observation and the nearest point in the shape file. Also used for polygon files, in which case the centroids are considered points. Default: *shapefile=NULL*.

`coormatrix` A matrix of geographic coordinates. The first column must be the longitude and the second column the latitude, both in degrees. In this case, *geoshape* calculates the distance between each observation and the nearest point in the matrix. The class of the object sent to *coormatrix* must actually be *matrix*. Default: *coormatrix=NULL*.

Details

Uses the *nncross* command from the *spatstat* package to calculate distances between the set of points given by *(longvar, latvar)* and the shape file provided by one of the *linefile*, *pointfile*, or *coormatrix* options. Only one of the three shape file options should be specified, and it is critical that the appropriate option is matched to the shape file. A polygon file is considered a point file for purposes of the *geoshape* command; in this case, *nncross* calculates distances to the polygon centroids. For the *coormatrix* option, it is critical that the object is of class *matrix*, the first column is longitude, and the second is latitude.

Value

Returns the calculated distances, in miles.

See Also

geodistance

Examples

```
data(matchdata)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1,]
lmat <- coordinates(cmap)
# Calculate distance between homes in matchdata and the census tract centroids
matchdata$dist1 <- geoshape(matchdata$longitude,matchdata$latitude,pointfile=cmap)
# Alternative method using coormatrix option
matchdata$dist2 <- geoshape(matchdata$longitude,matchdata$latitude,
  coormatrix=coordinates(cmap))

# measure distance from census tract centroids to Chicago city center
longitude <- lmat[,1]
latitude  <- lmat[,2]
cmat <- t(as.matrix(c(-87.627800, 41.881998)))
dcbd <- geoshape(longitude, latitude, coormatrix=cmat)
summary(dcbd)
```

gmmlogit *GMM Spatial Logit*

Description

Estimates a GMM logit model for a 0-1 dependent variable and an underlying latent variable of the form $Y^* = \rho WY^* + X\beta + u$

Usage

```
gmmlogit(form,inst=NULL,winst=NULL,wmat=NULL,shpfile,
  startb=NULL,startrho=0,blockid=0,cvcrit=.0001,data=NULL,silent=FALSE)
```

Arguments

form	Model formula
inst	List of instruments *not* to be pre-multiplied by *W*. Entered as *inst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
winst	List of instruments to be pre-multiplied by *W* before use. Entered as *winst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
wmat	Directly enter *wmat* rather than creating it from a shape file. Default: not specified. One of the *wmat* or *shpfile* options must be specified.
shpfile	Shape file to be used for creating the *W* matrix. Default: not specified. One of the *wmat* or *shpfile* options must be specified.
startb	Vector of starting values for *B*. Default: use estimates from *splogit*, the linearized version of the model. Specified as *startb=0.*
startrho	Vector of starting values for ρ. Default: use estimates from *splogit*, the linearized version of the model. Specified as *startrho=0.*
blockid	A variable identifying groups used to specify a block diagonal structure for the *W* matrix, e.g., *blockid=state* or *blockid=region*. Imposes that all elements outside of the blocks equal zero and then re-standardizes *W* such that the rows sum to one. By default, *blockid = 0*, and a block diagonal structure is *not* imposed.
cvcrit	Convergence criterion. Default: *cvcrit* = 0.0001.
data	A data frame containing the data. Default: use data in the current working directory.
silent	If *silent=T*, no output is printed

Details

The underlying latent variable for the model is $Y^* = \rho WY^* + X\beta + u$ or $Y^* = (I - \rho W)^{-1}(X\beta + u)$. The covariance matrix is $\sigma^2((I - \rho W)(I - \rho W)')^{-1}$, with σ^2 normalized to unity. Typical specifications imply heteroskedasticity, i.e., the diagonal elements of the covariance matrix, denoted by

σ_i^2, vary across observations. Heteroskedasticity makes standard logit estimates inconsistent. Letting $X_i^* = X_i/\sigma_i$ and $H = (I - \rho W)^{-1}X^*$, the logit probabilities implied by the latent variable are $p = exp(HB)/(1 + exp(HB))$ and the error term is $e_i = y_i - p_i$, where $y_i = 1$ if $Y_i^* > 0$ and $y_i = 0$ otherwise.

The GMM estimator chooses β and ρ to minimize $(y - p)'Z(Z'Z)^{-1}Z'(y - p)$, where *Z* is a matrix of instruments specified using the *inst* and *winst* options. Unless specified otherwise using the *startb* and *startrho* options, initial estimates are obtained using *splogit*, which implements the simple (and fast) linearized version of the GMM logit model proposed by Klier and McMillen (2008). Convergence is defined by $abs(change) < cvcrit$, where *change* is the gradient vector implied by applying a standard Gauss-Newton algorithm to the objective function. The covariance matrix (equation 3 in Klier-McMillen, 2008) is estimated using the *car* package.

Estimation can be very slow because each iteration requires the inversion of an *nxn* matrix. To speed up the estimation process and to reduce memory requirements, it may be desirable to impose a block diagonal structure on *W*. For example, it may be reasonable to impose that each state or region has its own error structure, with no correlation of errors across regions. The *blockid* option specifies a block diagonal structure such as *blockid=region*. The option leads the program to re-calculate the *W* matrix, imposing the block diagonal structure and re-normalizing the matrix to again have each row sum to one. If there are *G* groups, estimation requires *G* sub-matrices to be inverted rather than one *nxn* matrix, which greatly reduces memory requirements and significantly reduces the time required in estimation.

gmmlogit provides flexibility in specifying the list of instruments. By default, the instrument list includes *X* and *WX*, where *X* is the original explanatory variable list and *W* is the spatial weight matrix. It is also possible to directly specify the full instrument list or to include only a subset of the *X* variables in the list that is to be pre-multiplied by *W*.

Let *list1* and *list2* be user-provided lists of the form *list=~z1+z2*. The combinations of defaults (*NULL*) and lists for *inst* alter the final list of instruments as follows:

inst = NULL, *winst = NULL*: *Z = (X, WX)*
inst = list1, *winst = NULL*: *Z = list1*
inst = NULL, *winst = list2*: *Z = (X, W*list2)*
inst = list1, *winst = list2*: *Z = (list1, W*list2)*

Note that when *inst=list1* and *winst=NULL* it is up to the user to specify at least one variable in *list1* that is not also included in *X*.

Value

`coef`	Coefficient estimates
`se`	Standard error estimates

References

Klier, Thomas and Daniel P. McMillen, "Clustering of Auto Supplier Plants in the United States: Generalized Method of Moments Spatial Logit for Large Samples," *Journal of Business and Economic Statistics* 26 (2008), 460-471.

Pinkse, J. and M. E. Slade, "Contracting in Space: An Application of Spatial Statistics to Discrete-Choice Models," *Journal of Econometrics* 85 (1998), 125-154.

See Also

cparlogit

cparprobit

cparmlogit

gmmprobit

splogit

spprobit

spprobitml

Examples

```
# The example for "gmmprobit" applies directly after changing "gmmprobit" to "gmmlogit"
```

`gmmprobit` *GMM Spatial Probit*

Description

Estimates a GMM probit model for a 0-1 dependent variable and an underlying latent variable of the form $Y^* = \rho WY^* + X\beta + u$

Usage

```
gmmprobit(form,inst=NULL,winst=NULL,wmat=NULL,shpfile,
  startb=NULL,startrho=0,blockid=0,cvcrit=.0001,data=NULL,silent=FALSE)
```

Arguments

`form`	Model formula
`inst`	List of instruments *not* to be pre-multiplied by *W*. Entered as *inst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
`winst`	List of instruments to be pre-multiplied by *W* before use. Entered as *winst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.

wmat	Directly enter *wmat* rather than creating it from a shape file. Default: not specified. One of the *wmat* or *shpfile* options must be specified.
shpfile	Shape file to be used for creating the *W* matrix. Default: not specified. One of the *wmat* or *shpfile* options must be specified.
startb	Vector of starting values for *B*. Default: use estimates from *spprobit*, the linearized version of the model. Specified as *startb=0.*
startrho	Vector of starting values for ρ. Default: use estimates from *spprobit*, the linearized version of the model. Specified as *startrho=0.*
blockid	A variable identifying groups used to specify a block diagonal structure for the *W* matrix, e.g., *blockid=state* or *blockid=region*. Imposes that all elements outside of the blocks equal zero and then re-standardizes *W* such that the rows sum to one. By default, *blockid = 0*, and a block diagonal structure is *not* imposed.
cvcrit	Convergence criterion. Default: *cvcrit* = 0.0001.
data	A data frame containing the data. Default: use data in the current working directory.
silent	If *silent=T*, no output is printed

Details

The underlying latent variable for the model is $Y^* = \rho WY^* + X\beta + u$ or $Y^* = (I - \rho W)^{-1}(X\beta + u)$. The covariance matrix is $Euu' = \sigma^2((I - \rho W)(I - \rho W)')^{-1}$, with σ^2 normalized to unity. Typical specifications imply heteroskedasticity, i.e., the diagonal elements of Euu', denoted by σ_i^2, vary across observations. Heteroskedasticity makes standard probit estimates inconsistent. Letting $X_i^* = X_i/\sigma_i$ and $H = (I - \rho W)^{-1}X^*$, the probit probabilities implied by the latent variable are $p = \Phi(H\beta)$ and the generalized error term is $e = (y - p)\phi(H\beta)/(p(1 - p))$, where $y = 1$ if $Y^* > 0$ and $y = 0$ otherwise.

The GMM estimator chooses β and ρ to minimize $e'Z(Z'Z)^{-1}Z'e$, where *Z* is a matrix of instruments specified using the *inst* and *winst* options. Unless specified otherwise using the *startb* and *startrho* options, initial estimates are obtained using *spprobit*, which implements the simple (and fast) linearized version of the GMM probit model proposed by Klier and McMillen (2008). Convergence is defined by $abs(change) < cvcrit$, where *change* is the gradient vector implied by applying a standard Gauss-Newton algorithm to the objective function. The covariance matrix (equation 3 in Klier-McMillen, 2008) is estimated using the *car* package.

Estimation can be very slow because each iteration requires the inversion of an *nxn* matrix. To speed up the estimation process and to reduce memory requirements, it may be desirable to impose a block diagonal structure on *W*. For example, it may be reasonable to impose that each state or region has its own error structure, with no correlation of errors across regions. The *blockid* option specifies a block diagonal structure such as *blockid=region*. The option leads the program to re-calculate the *W* matrix, imposing the block diagonal structure and re-normalizing the matrix to again have each row sum to one. If there are *G* groups, estimation requires *G* sub-matrices to be inverted rather than one *nxn* matrix, which greatly reduces memory requirements and significantly reduces the time required in estimation.

gmmprobit provides flexibility in specifying the list of instruments. By default, the instrument list includes *X* and *WX*, where *X* is the original explanatory variable list and *W* is the spatial weight matrix. It is also possible to directly specify the full instrument list or to include only a subset of the *X* variables in the list that is to be pre-multiplied by *W*.

Let *list1* and *list2* be user-provided lists of the form *list=~z1+z2*. The combinations of defaults (*NULL*) and lists for *inst* alter the final list of instruments as follows:

inst = NULL, *winst = NULL*: *Z = (X, WX)*
inst = list1, *winst = NULL*: *Z = list1*
inst = NULL, *winst = list2*: *Z = (X, W*list2)*
inst = list1, *winst = list2*: *Z = (list1, W*list2)*

Note that when *inst=list1* and *winst=NULL* it is up to the user to specify at least one variable in *list1* that is not also included in *X*.

Value

`coef`	Coefficient estimates
`se`	Standard error estimates

References

Klier, Thomas and Daniel P. McMillen, "Clustering of Auto Supplier Plants in the United States: Generalized Method of Moments Spatial probit for Large Samples," *Journal of Business and Economic Statistics* 26 (2008), 460-471.

Pinkse, J. and M. E. Slade, "Contracting in Space: An Application of Spatial Statistics to Discrete-Choice Models," *Journal of Econometrics* 85 (1998), 125-154.

See Also

cparlogit

cparprobit

cparmlogit

gmmlogit

splogit

spprobit

Examples

```
set.seed(9947)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1&cmap$CAREA!="O'Hare",]
lmat <- coordinates(cmap)
```

```
dnorth <- geodistance(lmat[,1],lmat[,2], -87.627800,
41.881998, dcoor=TRUE)$dnorth
cmap <- cmap[dnorth>0,]
wmat <- makew(cmap)$wmat
n = nrow(wmat)
rho = .4
x <- runif(n,0,10)
ystar <- as.numeric(solve(diag(n) - rho*wmat)%*%(x + rnorm(n,0,2)))
y <- ystar>quantile(ystar,.4)
fit <- gmmprobit(y~x,  wmat=wmat)
```

---

kdensity *K-density functions for distances between geographic coordinates*

Description

Calculates *K*-density functions for lat-long coordinates. Calculates the distance, *d*, between every pair of observations and plots the density, $f(d_0)$, at a set of target distances, d_0. The kernel density functions are calculated using the *density* function.

Usage

```
kdensity(longitude,latitude,kilometer=FALSE,noplot=FALSE,
dmin=0,dmax=0,dlength=512,h=0,kern="gaussian",nsamp=0,
confint=TRUE,pval=.05)
```

Arguments

longitude	Longitude variable, in degrees.
latitude	Latitude variable, in degrees.
kilometer	If *kilometer = T*, measurements are in kilometers rather than miles. Default: *kilometer = F*.
noplot	If *noplot = T*, does not show the graph of the *K-density* function.
dmin	Minimum value for target distances. Default: *dmin*=0.
dmax	Maximum value for target distances. Default: *dmin* = max(distance), specified by setting *dmin*=0.
dlength	Number of target values for density calculations. Default: *dlength* = 512.
h	Bandwidth. Default: (.9*(quantile(distance,.75)-quantile(distance,.25))/1.34)*(n^(-.20)), where n = 2*length(dvect).
kern	Kernel. Default: "gaussian ". Other options from the *density* function are also available, including "epanechnikov", "rectangular", "triangular", "biweight", and "optcosine". The "cosine" kernel is translated to "optcosine".
nsamp	If *nsamp*>0, draws a random sample of lat-long pairs for calculations rather than the full data set. Can be much faster for large samples. Default: use full sample.
confint	If *TRUE*, adds local confidence intervals to the graph. Default: *confint=TRUE*.
pval	p-value for confidence intervals. Default: *pval=.05*.

Details

The *kdensity* function uses Silverman's (1986) reflection method to impose zero densities at negative densities. This method involves supplementing each distance observation with its negative value to form a pseudo data set with twice the original number of observations. The following commands are the core of the function:

```
dfit1 <- density(dvect,from=dmin,to=dmax,n=dlength,kernel=kern,bw=h)
dfit2 <- density(-dvect,from=dmin,to=dmax,n=dlength,kernel=kern,bw=h)
distance <- dfit1$x
dhat <- dfit1$y + dfit2$y
```

Local standard errors are calculated using the following asymptotic formula:

$(nh)^{-.5}(f(x)\int K^2(\psi)d\psi)^{.5}$

Value

`distance`	The vector of target distances.
`dhat`	The vector of densities for the target distances.
`dvect`	The full vector of distances between observation pairs. Length is n(n-1)/2.
`h`	The bandwidth.
`se`	The vector of standard errors.

References

Duranton, Gilles and Henry G. Overman, "Testing for Localisation using Microgeographic Data", *Review of Economic Studies* 72 (2005), 1077-1106.

Klier Thomas and Daniel P. McMillen, "Evolving Agglomeration in the U.S. Auto Industry," *Journal of Regional Science* 48 (2008), 245-267.

Silverman, A. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York (1986).

See Also

ksim

Examples

```
data(matchdata)
lmat <- cbind(matchdata$longitude,matchdata$latitude)
# Smaller sample to reduce computation time for example
set.seed(18493)
obs <- sample(seq(1,nrow(lmat)),400)
lmat <- lmat[obs,]
fit95 <- kdensity(lmat[,1],lmat[,2],noplot=FALSE)
```

ksim — *Estimates K-density functions with local or global confidence intervals for counter-factual locations*

Description

Calculates *K*-density functions for lat-long coordinates. Calculates the distance, *d*, between every pair of observations and plots the density, $f(d_0)$ at a set of target distances, d_0. Also uses the Duranton-Overman bootstrap method to construct local or global confidence intervals for the density of distances between pairs of observations if the same number of points were allocated across another set of possible locations.

Usage

```
ksim(long1,lat1,long2,lat2,kilometer=FALSE,noplot=FALSE,
  dmin=0,dmax=0,dlength=512,h=0,kern="gaussian",
  nsim=2000,nsamp=0,pval=.05,cglobal=FALSE)
```

Arguments

long1	Longitude variable, in degrees.
lat1	Latitude variable, in degrees.
long2	Longitude variable for counter-factual locations, in degrees.
lat2	Latitude variable for counter-factual locations, in degrees.
kilometer	If *kilometer* = *T*, measurements are in kilometers rather than miles. Default: *kilometer* = *F*.
noplot	If *noplot* = *T*, does not show the *K*-density graph.
dmin	Minimum value for target distances. Default: *dmin*=0.
dmax	Maximum value for target distances. Default: *dmin* = max(distance).
dlength	Number of target values for density calculations. Default: *dlength* = 512.
h	Bandwidth. Default: (.9*(quantile(distance,.75)-quantile(distance,.25))/1.34)*(n^(-.20)), where n = 2*length(dvect).
kern	Kernel. Default: "gaussian". Other options from the *density* function are also available, including "epanechnikov", "rectangular", "triangular", "biweight", "cosine", "optcosine".
nsim	Number of simulations for constructing the confidence intervals. Default: *nsim*=2000.
nsamp	If *nsamp*>0, uses a random sample of lat-long pairs for calculations rather than full data set. Takes random draws from *long1, lat1* pairs; the *long2, lat2* remain as specified by the user. Can be much faster for large samples. Default: use full sample.
pval	Significance level for confidence intervals. Default: *pval* = .05, i.e., a 95 percent confidence interval.
cglobal	If *cglobal*=*T*, calculates global confidence intervals. Default: *cglobal*=*F*, calculates local confidence intervals.

Details

Let *n* be the number of observations in the *long1, lat1* data set. *ksim* draws *n* observations from the *long2, lat2* pairs and re-calculates the *K*-density function using the new, simulated data set. The process is repeated *nsim* times, producing *nsim* bootstrap *K*-density functions. The local confidence interval treats each target distance as a separate observation, and calculates the confidence interval at each distance using the standard bootstrap percentile method. In contrast, the global confidence interval treats the full *K*-density function as an observations and shifts the interval outward at each data point until 95 percent of the density functions lie within the interval. Large values of *nsim* - perhaps greater than the default of 2000 - are necessary to get accurate global confidence intervals.

The *ksim* function is intended for cases where the counterfactual data set has more observations than the base, i.e., n2>n1. In this case, observations are drawn without replacement from the counterfactual data set. When the counterfactual data set has fewer observations than the base (i.e., n2<=n1), n1 observations are drawn with replacement from the counterfactual data set.

Duranton and Overman (2005) proposed this method for constructing global confidence intervals for *K*-density functions. See Klier and McMillen (2008) for a description of the procedures used here. See the description of the *kdensity* function for more details on the estimation procedure of the *K*-density functions.

Value

`distance`	The vector of target distances
`dhat`	The vector of densities for the target distances
`h`	The bandwidth
`local.lo`	The local confidence interval at each target distance, if calculated.
`local.hi`	The local confidence interval at each target distance, if calculated.
`global.lo`	The global confidence interval at each target distance, if calculated.
`global.hi`	The global confidence interval at each target distance, if calculated.

References

Duranton, Gilles and Henry G. Overman, "Testing for Localisation using Microgeographic Data", *Review of Economic Studies* 72 (2005), 1077-1106.

Klier Thomas and Daniel P. McMillen, "Evolving Agglomeration in the U.S. Auto Industry," *Journal of Regional Science* 48 (2008), 245-267.

Silverman, A. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York (1986).

See Also

kdensity

Examples

```
data(matchdata)
lmat <- cbind(matchdata$longitude,matchdata$latitude)
lmat1 <- lmat[matchdata$carea=="Rogers Park"|matchdata$carea=="Albany Park",]
```

```
lmat2 <- lmat[matchdata$carea!="Rogers Park"&matchdata$carea!="Albany Park",]
# smaller samples to reduce time for examples
set.seed(4941)
obs <- sample(seq(1,nrow(lmat1)),200)
lmat1 <- lmat1[obs,]
obs <- sample(seq(1,nrow(lmat2)),400)
lmat2 <- lmat2[obs,]

fit <- ksim(lmat1[,1],lmat1[,2],lmat2[,1],lmat2[,2],dmax=9,nsim=100,
  nsamp=100,noplot=TRUE,cglobal=FALSE)
ymin = min(fit$dhat,fit$local.lo)
ymax = max(fit$dhat,fit$local.hi)
plot(fit$distance, fit$dhat, xlab="Distance", ylab="Density", ylim = c(ymin,ymax),
  type="l", main="Albany Park & Rogers Park v. Other Areas")
lines(fit$distance, fit$local.lo, col="red")
lines(fit$distance, fit$local.hi, col="red")
```

`lwr` *Locally Weighted Regression*

Description

Estimates a model of the form *y = f(x)* using locally weighted regression. *x* can include either one or two variables. Returns estimated values, derivatives, and standard errors for both *f(x)* and *df(x)/dx*.

Usage

```
lwr(form,window=.25,bandwidth=0,kern="tcub",distance="Mahal",
  target=NULL,data=NULL)
```

Arguments

`form`	Model formula
`window`	Window size. Default: 0.25.
`bandwidth`	Bandwidth. Default: not used.
`kern`	Kernel weighting function. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
`distance`	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *lwr* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
`target`	If *target = NULL*, uses the *maketarget* command to form targets using the values specified for *window*, *bandwidth*, and *kern*. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
`data`	A data frame containing the data. Default: use data in the current working directory.

Details

The estimated value of *y* at a target value x_0 is the predicted value from a weighted least squares regression of *y* on $x - x_0$ with weights given by $K(\psi/h)$, where ψ is a measure of the distance between *x* and x_0 and *h* is the bandwidth or window.

When *x* includes a single variable, $\psi = x - x_0$. When *x* includes two variables, the method for specifying ψ depends on the *distance* option. If *distance="Mahal"* or *distance="Euclid"*, the *i*th row of the matrix *X* = (x1, x2) is transformed such that x_i = sqrt(*x_i * V * t(x_i)*). Under the "Mahal" option, *V* is the inverse of cov(*X*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*X*)). By reducing x from two dimensions to one, this transformation leads again to the simple kernel weighting function $K((x - x_0)/(sd(x) * h))$.

The great circle formula is used to define *K* when *distance = "Latlong"*; in this case, the explanatory variable list must be specified as *~latitude+longitude* (or *~lo+la* or *~lat+long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude are listed does not matter and the function only looks for the first two letters to determine which variable is latitude and which is the longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance = "Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option.

h is specified by the bandwidth or window options. The intercept, α, provides an estimate of *y* at x_0 and β provides an estimate of the slope, *dy/dx* at x_0. When *target="alldata"*, each data point in turn is used as a target point, x_0.

Since each estimate is a linear function of all *n* values for *y*, the full set of estimates takes the form $\hat{y} = LY$, where *L* is an *nxn* matrix. Loader (1999) suggests two measures of the number of degrees of freedom used in estimation, *df1 = tr(L)* and *df2 = tr(L'L)*, both of which are stored by *lwr*. The diagonal elements of *tr(L)* are stored in the array *infl*. Again following Loader (1999), the degrees of freedom correction used to estimate the error variance, $\hat{\sigma^2}$, is *df = 2*df1 - df2*. Let *e* represent the vector of residuals, $e = y - \hat{y}$. The estimated variance is $\hat{\sigma^2} = \sum_i e_i^2/(n - df)$. The covariance matrix is

$$\hat{\sigma}^2(\sum_{i=1}^{n} Z_i K(\psi_i/h) Z_i^\top)^{-1}(\sum_{i=1}^{n} Z_i (K(\psi_i/h))^2 Z_i^\top)(\sum_{i=1}^{n} Z_i K(\psi_i/h) Z_i^\top)^{-1}.$$

where $Z = (1\ x - x_0)$.

Estimation can be very slow when *target = "alldata"*. The *maketarget* command can be used to identify target points. The *smooth12* command is then used to interpolate the coefficient estimates, the standard errors, and the values used to form *df1* and *df2*.

h can be specified to be either a fixed bandwidth or a window size set to a percentage of the sample size. Optionally, the *lwrgrid* command can be used to specify a vector of values for *h* with *lwr* picking the one that minimizes a criterion function. In general, the *window* option will be preferable because it provides more accurate estimates in regions where *x* is relatively sparse.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\lvert z\rvert < 1)$
Triangular	"tria"	$(1 - \lvert z\rvert)I(\lvert z\rvert < 1)$

Epanechnikov	"epan"	$\frac{3}{4}(1-z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1-z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1-\|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1-z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target`	The target points for the original estimation of the function.
`ytarget`	The predicted values of *y* at the original target points.
`dtarget1`	The estimated derivatives *dy/dx1* at the target points.
`dtarget2`	The estimated derivatives *dy/dx2* at the target points. All zeros if the model has only one explanatory variable.
`ytarget.se`	Standard errors for the predicted values of *y* at the target points.
`dtarget1.se`	Standard errors for the derivatives *dy/dx1* at the target points.
`dtarget2.se`	Standard errors for the derivatives *dy/dx2* at the target points. All zeros if the model has only one explanatory variable.
`yhat`	The predicted values of *y* for the full data set.
`dhat1`	The estimated derivatives *dy/dx1* for the full data set.
`dhat2`	The estimated derivatives *dy/dx2* for the full data set. All zeros if the model has only one explanatory variable.
`yhat.se`	Standard errors for the predicted values of *y* for the full data set.
`dhat1.se`	Standard errors for the estimated derivatives *dy/dx1* for the full data set.
`dhat2.se`	Standard errors for the estimated derivatives *dy/dx2* for the full data set. All zeros if the model has only one explanatory variable.
`df1`	*tr(L)*, a measure of the degrees of freedom used in estimation.
`df2`	*tr(L'L)*, an alternative measure of the degrees of freedom used in estimation.
`sig2`	Estimated residual variance, *sig2 = rss/(n-2*df1+df2)*.
`cv`	Cross-validation measure. *cv = mean(((y-yhat)/(1-infl))^2)*, where *yhat* is vector of predicted values for *y* and *infl* is the vector of diagonal terms for *L*.
`gcv`	*gcv = n*(n*sig2)/((n-nreg)^2)*, where *sig2* is the estimated residual variance and *nreg = 2*df1 - df2*.
`infl`	A vector containing the diagonal elements of *L*.

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Loader, Clive. *Local Regression and Likelihood*. New York: Springer, 1999.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

McMillen, Daniel P., "Employment Densities, Spatial Autocorrelation, and Subcenters in Large Metropolitan Areas," *Journal of Regional Science* 44 (2004), 225-243.

McMillen, Daniel P. and John F. McDonald, "A Nonparametric Analysis of Employment Density in a Polycentric City," *Journal of Regional Science* 37 (1997), 591-612.

McMillen, Daniel P. and Christian Redfearn, "Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions," *Journal of Regional Science* 50 (2010), 712-733.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics*. New York: Cambridge University Press, 1999.

Silverman, A. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York (1986).

See Also

cparlwr

cubespline

fourier

lwrgrid

maketarget

semip

Examples

```
# 1. Monte Carlo data
n = 1000
x <- runif(n,0,2*pi)
x <- sort(x)
ybase <- x - .1*(x^2) + sin(x) - cos(x) -.5*sin(2*x) + .5*cos(2*x)
sig = sd(ybase)/2
y <- ybase + rnorm(n,0,sig)
par(ask=TRUE)
plot(x,y)
lines(x,ybase,col="red")
fit <- lwr(y~x, window=.15)
# plot 95% confidence intervals for predicted y
predse <- sqrt(fit$sig2 + fit$yhat.se^2)
lower <- fit$yhat + qnorm(.025)*predse
upper <- fit$yhat + qnorm(.975)*predse
plot(x, ybase, type="l", ylim=c(min(lower), max(upper)),
  main="Estimated Function", xlab="x", ylab="y")
lines(x, fit$yhat, col="red")
lines(x, lower, lty="dashed", col="red")
lines(x, upper, lty="dashed", col="red")
legend("topleft", c("Base", "Predicted", "95 Percent CI"),
 col=c("black", "red", "red"), lty=c("solid", "solid", "dashed"), lwd=1)

# plot 95%  confidence intervals for slopes
dxbase <- 1 - .2*x + cos(x) + sin(x) - cos(2*x) - sin(2*x)
lower <- fit$dhat1 + qnorm(.025)*fit$dhat1.se
```

```
upper <- fit$dhat1 + qnorm(.975)*fit$dhat1.se
plot(x, dxbase, type="l", ylim=c(min(lower), max(upper)),
  main="Estimated Slopes", xlab="x", ylab="y")
lines(x, fit$dhat1, col="red")
lines(x, lower, lty="dashed", col="red")
lines(x, upper, lty="dashed", col="red")
legend("topright", c("Base", "Predicted", "95 Percent CI"),
 col=c("black", "red", "red"),lty=c("solid", "solid", "dashed"), lwd=1)

# Derivative estimates with larger window size
fit <- lwr(y~x,window=.20)
lower <- fit$dhat1 + qnorm(.025)*fit$dhat1.se
upper <- fit$dhat1 + qnorm(.975)*fit$dhat1.se
plot(x, dxbase, type="l", ylim=c(min(lower), max(upper)),
  main="Estimated Slopes", xlab="x", ylab="y")
lines(x, fit$dhat1, col="red")
lines(x, lower, lty="dashed", col="red")
lines(x, upper, lty="dashed", col="red")
legend("topright", c("Base", "Predicted", "95 Percent CI"),
 col=c("black", "red", "red"), lty=c("solid", "solid", "dashed"), lwd=1)

## Not run:
#2. Population density data
library(RColorBrewer)

cook <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cook$obs <- seq(1:nrow(cook))
# measure distance to Chicago city center
lmat <- coordinates(cook)
cook$LONGITUDE <- lmat[,1]
cook$LATITUDE  <- lmat[,2]
cook$DCBD <- geodistance(longvar=cook$LONGITUDE,latvar=cook$LATITUDE,
  lotarget=-87.627800,latarget=41.881998,dcoor=FALSE)$dist
# population density = population/acres,  acres = square mile x 640
cook$LNDENS <- log(cook$POPULATION/(cook$AREA*640))
densdata <- data.frame(cook[cook$POPULATION>0,])
par(ask=TRUE)

# lndens = f(longitude, latitude), weights are function of straight-line distance
fit <- lwr(LNDENS~LONGITUDE+LATITUDE,  window=.10,
   distance="Latlong",data=densdata)
c(fit$df1, fit$df2, 2*fit$df1-fit$df2)
cook$lwrhat[densdata$obs] <- fit$yhat
brks <- seq(min(cook$lwrhat,na.rm=TRUE),max(cook$lwrhat,na.rm=TRUE),length=9)
spplot(cook,"lwrhat",at=brks,col.regions=rev(brewer.pal(9,"RdBu")),
   main="Log Density LWR Estimates")


## End(Not run)
```

`lwrgrid` *LWR Bandwidth or Window Selection*

Description

Finds the value of a user-provided array of window or bandwidth values that provides the lowest *cv* or *gcv* for an LWR model. Calls *lwr* and returns its full output for the chosen value of *h*.

Usage

```
lwrgrid(form,window=0,bandwidth=0,kern="tcub",method="gcv",print=TRUE,
  distance="Mahal",target=NULL,data=NULL)
```

Arguments

`form`	Model formula
`window`	Vector of possible window sizes. Default: none.
`bandwidth`	Vector of possible bandwidths. Default: none.
`method`	Specifies "gcv" or "cv" criterion function. Default: method="gcv".
`print`	If TRUE, prints *gcv* or *cv* values for each value of the window or bandwidth.
`kern`	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
`distance`	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *lwr* looks for the first two letters to determine which variable is latitude and which is longitude, so data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
`target`	If *target = NULL*, uses the *maketarget* command to form targets using *alpha = window*. In this case, the target points are found using a nearest neighbor procedure even if *bandwidth* > 0. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
`data`	A data frame containing the data. Default: use data in the current working directory

Value

`target`	The target points for the original estimation of the function.
`ytarget`	The predicted values of *y* at the original target points.
`dtarget1`	The estimated derivatives *dy/dx1* at the target points.
`dtarget2`	The estimated derivatives *dy/dx2* at the target points. All zeros if the model has only one explanatory variable.
`ytarget.se`	Standard errors for the predicted values of *y* at the target points.

`dtarget1.se`	Standard errors for the derivatives *dy/dx1* at the target points.
`dtarget2.se`	Standard errors for the derivatives *dy/dx2* at the target points. All zeros if the model has only one explanatory variable.
`yhat`	The predicted values of *y* for the full data set.
`dhat1`	The estimated derivatives *dy/dx1* for the full data set.
`dhat2`	The estimated derivatives *dy/dx2* for the full data set. All zeros if the model has only one explanatory variable.
`yhat.se`	Standard errors for the predicted values of *y* for the full data set.
`dhat1.se`	Standard errors for the estimated derivatives *dy/dx1* for the full data set.
`dhat2.se`	Standard errors for the estimated derivatives *dy/dx2* for the full data set. All zeros if the model has only one explanatory variable.
`df1`	*tr(L)*, a measure of the degrees of freedom used in estimation.
`df2`	*tr(L'L)*, an alternative measure of the degrees of freedom used in estimation.
`sig2`	Estimated residual variance, *sig2 = rss/(n-2*df1+df2)*.
`cv`	Cross-validation measure. *cv = mean(((y-yhat)/(1-infl))^2)*, where *yhat* is vector of predicted values for *y* and *infl* is the vector of diagonal terms for *L*.
`gcv`	*gcv = n*(n*sig2)/((n-nreg)^2)*, where *sig2* is the estimated residual variance and *nreg = 2*df1 - df2*.
`infl`	A vector containing the diagonal elements of *L*.
`minh`	Value of window or bandwidth the minimizes the criterion function.

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Loader, Clive. *Local Regression and Likelihood*. New York: Springer, 1999.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

McMillen, Daniel P., "Employment Densities, Spatial Autocorrelation, and Subcenters in Large Metropolitan Areas," *Journal of Regional Science* 44 (2004), 225-243.

McMillen, Daniel P. and John F. McDonald, "A Nonparametric Analysis of Employment Density in a Polycentric City," *Journal of Regional Science* 37 (1997), 591-612.

McMillen, Daniel P. and Christian Redfearn, "Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions," *Journal of Regional Science* 50 (2010), 712-733.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics*. New York: Cambridge University Press, 1999.

Silverman, A. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, New York (1986).

See Also

lwr

maketarget

Examples

```
par(ask=TRUE)
n = 1000
z1 <- runif(n,0,2*pi)
z1 <- sort(z1)
z2 <- runif(n,0,2*pi)
o1 <- order(z1)
o2 <- order(z2)
ybase1 <-  z1 - .1*(z1^2) + sin(z1) - cos(z1) - .5*sin(2*z1) + .5*cos(2*z1)
ybase2 <- -z2 + .1*(z2^2) - sin(z2) + cos(z2) + .5*sin(2*z2) - .5*cos(2*z2)
ybase <- ybase1+ybase2
sig = sd(ybase)/2
y <- ybase + rnorm(n,0,sig)
summary(lm(y~ybase))

# Single variable estimation
fit1 <- lwrgrid(y~z1,window=seq(.10,.30,.05))
c(fit1$df1,fit1$df2,2*fit1$df1-fit1$df2)
plot(z1[o1],ybase1[o1],type="l",ylim=c(min(ybase1,fit1$yhat),max(ybase1,fit1$yhat)),
  xlab="z1",ylab="y")
# Make predicted and actual values have the same means
fit1$yhat <- fit1$yhat - mean(fit1$yhat) + mean(ybase1)
lines(z1[o1],fit1$yhat[o1], col="red")
legend("topright", c("Base", "LWR"), col=c("black","red"),lwd=1)
fit2 <- lwrgrid(y~z2,window=seq(.10,.90,.10))
fit2$yhat <- fit2$yhat - mean(fit2$yhat) + mean(ybase2)
c(fit2$df1,fit2$df2,2*fit2$df1-fit2$df2)
plot(z2[o2],ybase2[o2],type="l",ylim=c(min(ybase2,fit2$yhat),max(ybase2,fit2$yhat)),
    xlab="z1",ylab="y")
lines(z2[o2],fit2$yhat[o2], col="red")
legend("topright", c("Base", "LWR"), col=c("black","red"),lwd=1)

#both variables
fit3 <- lwrgrid(y~z1+z2,window=seq(.03,.09,.02))
yhat1 <- fit3$yhat - mean(fit3$yhat) + mean(ybase1)
plot(z1[o1],yhat1[o1], xlab="z1",ylab="y")
lines(z1[o1],ybase1[o1],col="red")
yhat2 <- fit3$yhat - mean(fit3$yhat) + mean(ybase2)
plot(z2[o2],yhat2[o2], xlab="z2",ylab="y")
lines(z2[o2],ybase2[o2],col="red")
```

`maketarget` *Target Points for Nonparametric Models*

Description

Identifies target points at which to evaluate nonparametric models with one or two explanatory variables.

Usage

```
maketarget(form,window=.25,bandwidth=0,kern="tcub",actualobs=FALSE,data=NULL)
```

Arguments

form	The model formula to be used to determine target points. The dependent variable is irrelevant, so form=y~x and form=~x are equivalent. No more than two explanatory variables should be listed.
window	The window size.
bandwidth	The bandwidth. The *window* option is ignored if *bandwidth* >0.
kern	The kernel weight function.
actualobs	If FALSE, the points identifed as targets are not constrained to be actual data points. Specifying *actualobs* = *TRUE* produces target points that are drawn from the original data matrix implied by *form*.
data	A data frame containing the data. Default: use data in the current working directory

Details

The *maketarget* function uses the *locfit* package's adaptive decision tree approach to identify target locations. If *actualobs* = *TRUE*, the output of *maketarget* is set of actual data points closest to these target locations, along with the convex hull identified by the *chull* command. The variable *obs* can be used to indicate the target observations for *actualobs* models. This *actualobs* = *TRUE* option is required by the following commands: lwr, lwrgrid, cparlwr, cparlwrgrid, and semip.

Value

target	The set of target points.
obs	If *actualobs* = *TRUE*, *obs* is the list of observation numbers in the original data set from which the target points are drawn. *obs* = *NULL* if *actualobs* = *FALSE*.

References

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999. Section 12.2.

See Also

cparlogit cparlwr cparlwrgrid cparmlogit cparprobit lwr lwrgrid qregcpar qreglwr semip

Examples

```
data(cookdata)
target <- maketarget(~LONGITUDE+LATITUDE,window=.25,data=cookdata)$target
```

makew *Calculation of spatial weight matrices*

Description

Constructs a spatial weight matrix from a shape file or a matrix of geographic coordinates

Usage

```
makew(shpfile=NULL,coormat=NULL,method="queen",knum=10,
  ringdist=.25,kern="tcub",window=.10,eigenvalues=FALSE)
```

Arguments

shpfile	A shape file.
coormat	A matrix of geographic coordinates. The first column should be the longitude and the second latitude, in degrees.
method	Options using shape files to identify first-order contiguity are "queen" and "rook". Options requiring a matrix of geographic coordinates are "knear", "ring", and "kernel". The coordinate matrix can be inputted directly with the *coormat* option, or it can be calculated directly from the shape file. The shape file takes precedence if both *shapefile* and *coormat* are specified.
knum	The number of nearest neighbors for the *knear* option. Default: *knum* = 10
ringdist	The maximum distance for the *ring* option. Default: *ringdist* = .25
kern	The kernel function for the *kernel* option. Options include *"rect", "tria", "epan", "bisq", "tcub"* and *"trwt"*.
window	Window size for the *kernel* option. Default: *window* = .25
eigenvalues	If *TRUE*, calculates eigenvalues. Default: *eigenvalues=FALSE*.

Details

If *method=rook* or *method=queen*, an nxn contiguity matrix is defined using the *spdep* package. A *queen* definition of contiguity means that two tracts are defined as contiguous if they share at least one point. A better definition of *rook* might be "non-queen": two tracts are defined as contiguous if they share two or more points. The *rook* and *queen* options require a shape file. The contiguity matrix is row-normalized to form the weight matrix.

If *method=knear*, the k nearest neighbors are each given a weight of 1/k to form W. The calculations are made using the *spdep* package. Either a shape file or a matrix of geographic coordinates can be provided to the *knear* option.

If *method=ring*, each observation within a distance of *ringdist* from observation i is given equal weight in row i. More distant observations receive a weight of zero.

If *method=kernel*, a kernel weight function is used to define W, with the window size determined by the *window* option. The kernel weight function is defined by the *kern* option. The weights

are $W_{ij} = K(d_{ij}/h)/h$ for $d_{ij} < h$ and $W_{ij} = 0$ for i = j and $d_{ij} > h$. The matrix is then row-normalized.

Eigenvalues are returned if the *eigenvalues=T* is specified.

Value

wmat	The nxn spatial weight matrix. The matrix is row-normalized.
eigvar	The eigvenvalues of *wmat*. Calculated if *eigenvalues=TRUE*.

See Also

sarml

qregspiv

Examples

```
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$POPULATION>0&cmap$AREA>0,]
cmap <- cmap[cmap$CHICAGO==1&cmap$CAREA!="O'Hare",]
lmat <- coordinates(cmap)
fit <- makew(shpfile=cmap,method="queen")
# fit <- makew(coormat=lmat,method="ring",ringdist=1)
```

---

matchdata *Matched samples of house sales in Chicago for 1995 and 2005*

Description

Sales prices, structural characteristics, and location variables for 1602 single-family homes in the City of Chicago in 1995 and a matched sample of 1602 homes in 2005.

Usage

```
data(matchdata)
```

Format

A data frame with 3204 observations on the following 18 variables.

`year` Year of sale, 1995 or 2005

`lnland` Log of land area in square feet

`lnbldg` Log of building area in square feet

`rooms` Number of rooms

`bedrooms` Number of bedrooms

`bathrooms` Number of bathrooms

`centair` Home has central air conditioning

`fireplace` Home has one or more fireplaces

`brick` Brick or brick/frame construction

`garage1` Garage, 1 car

`garage2` Garage, 2+ cars

`dcbd` Distance from the central business district or "CBD" - the traditional center of Chicago at the intersection of State and Madison Streets, at approximately -87.627800 longitude and 41.881998 latitude

`rr` Within .25 miles of a rail line

`yrbuilt` Year the home was built

`carea` a factor with levels. Community area, a traditional definition of neighborhood in Chicago.

`latitude` Latitude in degrees

`longitude` Longitude in degrees

`lnprice` Log of sales price

Details

Includes all sales of single-family homes on the Far North Side of Chicago listed in the cleaned Illinois Department of Revenue file for 1995. A matched sample is created from comparable 2005 sales using the MatchIt package. Matches are created based on propensity scores estimated using a logit model for the probability that a home sold in 2005 rather than 1995. The commands used to create the matched sample are the following:

```
hedonic$carea <- as.factor(hedonic$cname)
m.out <- matchit(y~lnland + lnbldg + rooms + bedrooms + bathrooms + centair + fireplace +
brick + garage1 + garage2 + dcbd + elstop + lake + rr + yrbuilt + carea + latitude + longitude,
data=hedonic,method="nearest",discard="both")
mdata <- match.data(m.out)
attach(mdata)
matchdata <- data.frame(year, lnland, lnbldg, rooms, bedrooms, bathrooms, centair, fireplace, brick,
garage1, garage2, dcbd, rr, yrbuilt, carea, latitude, longitude, lnprice)
```

The elstop and lake variables, which are not included here, indicate whether a home is within .25 miles of and EL stop and within .5 miles of Lake Michigan.

Source

Daniel McMillen. Sales data were provided originally by the Illinois Department of Revenue. Structural characteristics are drawn from the 1997 assessment file from the Cook County Assessor's Office.

References

Deng, Yongheng, Sing Tien Foo, and Daniel P. McMillen, "Private Residential Price Indices in Singapore," *Regional Science and Urban Economics*, 42 (2012), 485-494.

Ho, D., Imai, K., King, G, Stuart, E., "Matching as Nonparametric Preprocessing for Reducing Model Dependence in Parametric Causal Inference," *Political Analysis* 15 (2007), 199-236.

Ho, D., Imai, K., King, G, Stuart, E., "MatchIt: Nonparametric preprocessing for parametric causal inference," *Journal of Statistical Software* 42 (2011), 1-28..

McMillen, Daniel P., "Repeat Sales as a Matching Estimator," *Real Estate Economics* 40 (2012), 743-771.

Examples

```
data(matchdata)
matchdata$year05 <- matchdata$year==2005
matchdata$age <- matchdata$year - matchdata$yrbuilt
fit <- lm(lnprice~lnland+lnbldg+rooms+bedrooms+bathrooms+centair+fireplace+brick+
  garage1+garage2+dcbd+rr+age+year05+factor(carea), data=matchdata)
summary(fit)
```

`matchmahal` *Matched sample data frame based on mahalanobis distances*

Description

Creates a matched sample data frame based on mahalanobis distances

Usage

```
matchmahal(form,data=NULL,discard="none",
  distance="logit",m.order="none",nclose=0,ytreat=1)
```

Arguments

`form`	Model formula
`data`	A data frame containing the data. Default: use data in the current working directory
`discard`	Observations to be discarded based on the propensity score. If *discard* = "control", only control observations are discarded. If *discard* = "treat", only treatment observations are discarded. If *discard* = "both", both control and treatment observations are deleted. Default: *discard* = "none"; no options are discarded and propensity scores are not estimated.
`distance`	The link formula to be passed on to the glm command if discard = "control", "treat", or "both"; default = "logit"

`m.order` Order by which estimated distances are sorted before starting the matching process. Options: "decreasing", "increasing", "random", and "none". As the "decreasing" and "increasing" options are based on propensity scores, they are only applicable when discard = "control", "treat", or "both".

`nclose` If *nclose*>0, sorts the matched observations by the distance measure and chooses the *nclose* matches with the smallest distances.

`ytreat` The value of the dependent variable for the treatment group. Default: *ytreat* = 1. Constructs matched samples for all other values of the dependent variable. If *discard*="treat" or *discard*="both", only treatment observations that were discarded for *every* control value of the dependent variable are omitted from the final data set.

Details

Creates a matched sample data set by matching each treatment variable to the closest control variable based on mahalanobis distances. Like *matchprop*, *matchmahal* is particularly useful for creating a series of matched sample data sets over time relative to a base time period.

Let *X1* be the matrix of explanatory variables for the treatment observations and let *X2* be the comparable matrix for the control observations. The mahalanobis measure of distance between the *i*th row of *X1* and all control observations is $d_i = mahalanobis(X2, X1[i,], cov(rbind(X2, X1)))$. The first observation of *X1* is matched with the closest observation in *X2* based on this distance measure. The row is then removed from *X2* and the second observation of *X1* is matched with the closest of the remaining control observations. The process is repeated until there are no more observations left in one of the matrices.

By default, *matchprop* matches every treatment observation with a control observation. If the number of treatment observations (n1) is less than the number of control observations (n2), then the first n2 treatment observations will be in the final matched sample data set. By default, the observations are matched in the order in which they appear in the original data set. Alternatively, the observations can be matched in random order by specifying *m.order* = "random".

The *distance* option allows the user to specify a metric by which observations are determined to be outside the probability support. The same options are available as in the *matchprop* command. The natural one is *distance* = "mahal" combined with *discard* = "control", "treat" or "both" and *m.order* = "increasing", "decreasing", or "random". Other options are listed in the documentation for the *matchprop* command, e.g., *distance* = "logit" or "probit". Any of the these *distance* options produces a propensity score, *p*. When *distance* = "mahal", the propensity score is the mahalanobis distance of each observation from the vector of means. The *discard* option determines how observations are handled that are outside the probability support. For example, if the treatment is set to *ytreat* = 1 and the alternative value of the dependent variable is *y* = 2, then:

discard = "control": observations with p[y==2]<min(p[y==1]) are discarded from the y==2 sample

discard = "treat": observations with p[y==1]>max(p[y==2]) are discarded from the y==1 sample

discard = "both": both sets of observations are deleted

If *discard* = "treat" or "both" and the dependent variable has more than two values, a different set of treatment observations may be discarded as being outside the support of the two propensity measures. Only treatment observations that are rejected by *both* models will end up being omitted from the final data set.

Value

Returns the matched sample data frame. Adds the following variables to the data set:

origobs: The observation number in the original data set

matchobs: The observation number in the matched data set to which the observation is matched. *matchobs* refers to the observation's number in the original data set, i.e., to the variable *origobs*.

Note: If the original data set includes variables named *origobs* and *matchobs*, they will be overwritten by the variables produced by *matchmahal*.

References

Deng, Yongheng, Sing Tien Foo, and Daniel P. McMillen, "Private Residential Price Indices in Singapore," *Regional Science and Urban Economics*, 42 (2012), 485-494.

Ho, D., Imai, K., King, G, Stuart, E., "Matching as Nonparametric Preprocessing for Reducing Model Dependence in Parametric Causal Inference," *Political Analysis* 15 (2007), 199-236.

Ho, D., Imai, K., King, G, Stuart, E., "MatchIt: Nonparametric preprocessing for parametric causal inference," *Journal of Statistical Software* 42 (2011), 1-28..

McMillen, Daniel P., "Repeat Sales as a Matching Estimator," *Real Estate Economics* 40 (2012), 743-771.

See Also

matchprop

matchqreg

Examples

```
set.seed(189)
n = 1000
x <- rnorm(n)
x <- sort(x)
y <- x*1 + rnorm(n, 0, sd(x)/2)
y <- ifelse(y>0,1,0)
table(y)
fit <- matchmahal(y~x,ytreat=1)
table(fit$y)
```

---

matchprop *Matched sample data frame based on propensity scores*

Description

Creates a matched sample data frame based on propensity scores

Usage

```
matchprop(form,data=NULL,distance="logit",discard="both",
  reestimate="FALSE",m.order="none",nclose=0,ytreat=1)
```

Arguments

form	Model formula
data	A data frame containing the data. Default: use data in the current working directory
distance	The link formula to be passed on to the glm command – usually "probit" or "logit", but other standard options also work.
discard	Observations to be discarded based on the propensity score or the value of the mahalanobis distance measure if *distance*="mahal". If *discard* = "control", only control observations are discarded. If *discard* = "treat", only treatment observations are discarded. If *discard* = "both", both control and treatment observations are deleted.
reestimate	If *reestimate*=TRUE, the propensity score is reestimated after observations are discarded
m.order	Order by which estimated distances are sorted before starting the matching process. Options: "decreasing", "increasing", "random", and "none".
nclose	If *nclose*>0, sorts the matched observations by the distance measure and chooses the *nclose* matches with the smallest distances.
ytreat	The value of the dependent variable for the treatment group. Default: *ytreat* = 1. Constructs matched samples for all other values of the dependent variable. If *discard*="treat" or *discard*="both", only treatment observations that were discarded for *every* control value of the dependent variable are omitted from the final data set.

Details

Creates a matched sample data set using procedures based on the *MatchIt* program. *MatchIt*'s routines are generally preferable for creating a single matched data set, although by doing less the *matchprop* command is somewhat faster than *MatchIt*. *matchprop* is particularly useful for creating a series of matched sample data sets over time relative to a base time period.

Unless *distance* = "mahal", the *glm* command is used to estimate the propensity scores using a series of discrete choice models for the probability, *p*, that the dependent variable equals *ytreat* rather than each alternative value of the dependent variable. The default link function is *distance* = *"logit"*. Alternative link functions are specified using the *distance* option. Links include the standard ones for a *glm* model with *family* = *binomial*, e.g., "probit", "cauchit", "log", and "cloglog".

If *mahal*= T, *matchprop* implements *MatchIt's* version of mahalanobis matching. Letting *X* be the matrix of explanatory variables specified in *form*, the mahalanobis measure of distance from the vector of mean values is $p = mahalanobis(X, colMeans(X), cov(X))$. Although this version of mahalanobis matching is fast, it may not be the best way to construct matches because it treats observations that are above and below the mean symmetrically. For example, if *X* is a single variable with $mean(X)$ = .5 and $var(X)$ = 1, mahalanobis matching treats *X* =.3 and *X* = .7 the same: mahalanobis(.3,.5,1) = .04 and mahalanobis(.7,.5,.1) = .04. The function *matchmahal* is slower but

generally preferable for mahalanobis matching because it pairs each treatment observation with the closest control observation, i.e., $min(mahalanobis(X0, X1[i,], cov(X)))$, where *X0* is the matrix of explanatory variables for the control observations, *X1* is the matrix for the treatment observations, *X* is the pooled explanatory variable matrix, and *i* is the target treatment observation.

To illustrate how *matchprop* constructs matched samples, suppose that the dependent variable takes on three values, *y* = 1, 2, 3, and assume that *y* = 1 is the treatment group. First, the *y* = 1 and *y* = 2 observations are pooled and a propensity score *p* is constructed by, e.g., estimating a logit model for the probability that *y* = 1 rather than 2. Unless *m.order* = "none", the data frame is then sorted by *p* – from largest to smallest if *m.order* = "largest", from smallest to largest if *m.order* = "smallest", and randomly if *m.order* = "random". The first treatment observation is then paired with the closest control observation, the second treatment observation is paired with the closest of the remaining control observations, and so on until the last observation is reached for one of the groups. No control observation is matched to more than one treatment observation, and only pairwise matching is supported using *matchprop*. The process is then repeated using the *y* = 1 and *y* = 3 observations. If the number of treatment observations is n1, the final data set will have roughly 3*n1 observations – the n1 treatment observations and n1 observations each from the *y* = 2 and *y* = 3 observations. The exact number of observations will differ depending on how observations are treated by the *discard* option, and there will be fewer than n1 observations for, e.g., group 2 if n2<n1.

The *discard* option determines how observations are handled that are outside the probability support. In the above example, let p be the propensity score for the logit model for the probability that *y* = 1 rather than 2. If *discard* = "control", observations with p[y==2]<min(p[y==1]) are discarded from the y==2 sample. If *discard* = "treat", observations with p[y==1]>max(p[y==2]) are discarded from the y==1 sample. If *discard* = "both", both sets of observations are deleted. The process is then repeated for the *y* = 1 and *y* = 3 observations. If *discard* = "treat" or "both", a different set of treatment observations may be discarded as being outside the support of the two propensity measures. Only treatment observations that are rejected by *both* models will end up being omitted from the final data set.

If *reestimate* = T, the propensity scores are reestimated after any observations are discarded. Otherwise, matches are based on the original propensity scores.

Value

Returns the matched sample data frame. Adds the following variables to the data set:

origobs: The observation number in the original data set

matchobs: The observation number in the matched data set to which the observation is matched. *matchobs* refers to the observation's number in the original data set, i.e., to the variable *origobs*.

Note: If the original data set includes variables named *origobs* and *matchobs*, they will be overwritten by the variables produced by *matchprop*.

References

Deng, Yongheng, Sing Tien Foo, and Daniel P. McMillen, "Private Residential Price Indices in Singapore," *Regional Science and Urban Economics*, 42 (2012), 485-494.

Ho, D., Imai, K., King, G, Stuart, E., "Matching as Nonparametric Preprocessing for Reducing Model Dependence in Parametric Causal Inference," *Political Analysis* 15 (2007), 199-236.

Ho, D., Imai, K., King, G, Stuart, E., "MatchIt: Nonparametric preprocessing for parametric causal inference," *Journal of Statistical Software* 42 (2011), 1-28..

McMillen, Daniel P., "Repeat Sales as a Matching Estimator," *Real Estate Economics* 40 (2012), 743-771.

See Also

matchmahal

matchqreg

Examples

```
set.seed(189)
n = 1000
x <- rnorm(n)
x <- sort(x)
y <- x*1 + rnorm(n, 0, sd(x)/2)
y <- ifelse(y>0,1,0)
table(y)
fit <- matchprop(y~x,m.order="largest",ytreat=1)
table(fit$y)
```

---

matchqreg *Sample quantiles and means over time for a matched sample data set*

Description

Calculates and graphs sample means and quantiles over time. Intended for but not limited to a data set constructed with *matchprop* or *matchmahal*

Usage

```
matchqreg(form,taumat=c(.10,.25,.50,.75,.90), qreglwr.smooth=TRUE,
  window=.50,bandwidth=0,kern="tcub", alldata=FALSE,
  graph.yhat=TRUE,graph.mean=TRUE,data)
```

Arguments

form	A formula of the type *y~x*, where *x* represents time.
taumat	Vector of quantiles. Default: *taumat*=c(.10, .25, .50, .75, .90).
qreglwr.smooth	If *qreglwr.smooth=T*, uses *qreglwr* to smooth the quantile series. If *qreglwr.smooth=F*, calculates period by period quantiles.
window	Window size to be passed to *qreglwr* if *qreglwr.smooth=T*. Default: 0.50.
bandwidth	Bandwidth to be passed to *qreglwr* if *qreglwr.smooth=T*. Default: 0, i.e., not used.

kern	Kernel weighting function to be passed to *qreglwr* if *qreglwr.smooth=T*. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
alldata	Indicates how the *alldata* option should be treated for *qreglwr* if *qreglwr.smooth=T*. Default: *alldata=F*
graph.yhat	If *graph.yhat=T*, graphs the series of quantile lines. Default: *graph.yhat=T*.
graph.mean	If *graph.mean=T*, graphs the means over time. Default: *graph.yhat=T*.
data	A data frame containing the data. Default: use data in the current working directory.

Details

Calculates means and quantiles of *y* for each time period present in the variable on the right hand side of the model formula. The quantiles can be varied with the *taumat* option. If *qreglwr.smooth=T*, matchqreg uses the qreglwr command to smooth the quantile lines and stores the results in the matrix *yhat*. The unsmoothed, actual quantile values are stored in *yhat* if *qreglwr.smooth=F*. The *window, bandwidth, kern,* and *alldata* options are passed on to *qreglwr* if *qreglwr.smooth=T*.

Although *matchqreg* is meant to follow the *matchprop* or *matchmahal* command, it can be applied to any data set.

Value

yhat	Matrix of quantiles for y; actual quantiles if *qreglwr.smooth=F* and smoothed values if *qreglowr.smooth=T*. Rows represent time periods and columns represent quantiles.
ymean	Average value of y for each time period.
timevect	Vector of target quantile values.

References

Deng, Yongheng, Sing Tien Foo, and Daniel P. McMillen, "Private Residential Price Indices in Singapore," *Regional Science and Urban Economics*, 42 (2012), 485-494.

Ho, D., Imai, K., King, G, Stuart, E., "Matching as Nonparametric Preprocessing for Reducing Model Dependence in Parametric Causal Inference," *Political Analysis* 15 (2007), 199-236.

Ho, D., Imai, K., King, G, Stuart, E., "MatchIt: Nonparametric preprocessing for parametric causal inference," *Journal of Statistical Software* 42 (2011), 1-28..

McMillen, Daniel P., "Repeat Sales as a Matching Estimator," *Real Estate Economics* 40 (2012), 743-771.

See Also

matchmahal

matchprop

qreglwr

Examples

```
set.seed(189)
n = 500
# sale dates range from 0-10
# mean and variance of x increase over time, from 1 to 2
# price index for y increases from 0 to 1
timesale <- array(0,dim=n)
x <- rnorm(n,0,1)
for (j in seq(1,10)) {
  timesale <- c(timesale, array(j, dim=n))
  x <- c(x, rnorm(n,j/10,1+j/10))
}
n = length(x)
y <- x*1 + timesale/10 + rnorm(n, 0, sd(x)/2)
fit <- lm(y~x+factor(timesale))
summary(fit)
heddata <- data.frame(y,x,timesale)
summary(heddata)

par(ask=TRUE)
matchdata <- matchprop(timesale~x,data=heddata,ytreat=0,
  distance="logit",discard="both")
table(matchdata$timesale)
fit <- matchqreg(y~timesale,qreglwr.smooth=FALSE,
  graph.yhat=TRUE,graph.mean=TRUE,data=matchdata)
```

---

qregbmat *Quantile Regression for Multiple Quantiles*

Description

Returns estimated coefficients from a series of quantile regressions.

Usage

```
qregbmat(form,taumat=seq(.10,.90,.10),graphb=TRUE,graph.factor=FALSE,
 data=NULL)
```

Arguments

form	Model formula
taumat	Vector of target quantiles. Default: *taumat=seq(.10,.90,.10)*
graphb	If *graphb=TRUE*, prints graphs of the coefficient estimates. Default: *graphb=TRUE*.
graph.factor	If *graph.factor=TRUE* and *graphb=TRUE*, prints graphs of the coefficient estimates for any factor variables. Default: *graph.factor=TRUE*.
data	A data frame containing the data. Default: use data in the current working directory.

Details

Estimates a series of quantile regressions using the *quantreg* packages. The quantiles are listed in *taumat*. The *qregbmat* command is intended primarily as a first stage before the *qregsim1* or *qregsim2* commands.

Value

Returns the length(taumat) x k matrix of estimated coefficients, where k is the number of explanatory variables.

References

Koenker, Roger. *Quantile Regression*. New York: Cambridge University Press, 2005.

See Also

qregsim1

qregsim2

qregcpar

qreglwr

Examples

```
par(ask=TRUE)
data(matchdata)
matchdata$age <- matchdata$year - matchdata$yrbuilt
bmat <- qregbmat(lnprice~lnland+lnbldg+age+factor(year), data=matchdata,
  graph.factor=TRUE)
summary(bmat)
```

---

qregcdf *Nonparametric quantiles based on conditional CDF functions*

Description

Estimates conditional quantile functions based on nonparametric conditional CDF functions

Usage

```
qregcdf(form,taumat=c(.10,.25,.50,.75,.90),hx=0,hy=0,nx=20,ny=100,
  targetx=0,targety=0,graph.target=FALSE,graph.yhat=FALSE, data=NULL)
```

Arguments

form	Model formula
taumat	Vector of quantiles. Default: taumat=c(.10, .25, .50, .75, .90)
hx	Bandwidth for *x* in $K((X_i-x)/hx)$. Default: *hx* = 1.06*min(sd(x), (quantile(x,.75)-quantile(x,.25))/1.349)*(n^-.2)
hy	Bandwidth for *y* in $\Phi((y-Y_i)/hy)$ Default: *hy* = 1.06*min(sd(y), (quantile(y,.75)-quantile(y,.25))/1.349)*(n^(-.2))
nx	Number of target points for *x* if using an evenly spaced grid. Default is *nx* = 20. If *nx*>0, then *targetx* <- seq(min(x),max(x),length=nx)
ny	Number of target points for *y* if using an evenly spaced grid. Default is *ny* = 200. If *ny*>0, then $targety < -seq(min(y), max(y), length = ny)$
targetx	Vector of user-provided target values for *x*. An alternative to using *nx* to specify a uniformly spaced grid. Default: not specified.
targety	Vector of user-provided target values for *y*. An alternative to using *nxy* to specify a uniformly spaced grid. Default: not specified.
graph.target	If *graph.target=T*, graph of results is produced based on target values. Default: *graph.target=F*.
graph.yhat	If *graph.yhat=T*, graph of results is produced based on interpolations to actual values of *x* and *y*. Default: *graph.yhat=F*.
data	A data frame containing the data. Default: use data in the current working directory

Details

Following Li and Racine (2007), equation 6.3, a smoothed version of the conditional CDF of *y* given a value of *x* can be written:

$$F(y|x) = \frac{\frac{1}{n}\sum_i \Phi(\frac{y-y_i}{h_y}) h_x^{-1} K(\frac{X_i-x}{h_x})}{h_x^{-1}\sum_i K(\frac{X_i-x}{h_x})}$$

The estimation procedure begins by evaluating this expression at each point in the grid determined by the values of *target.x* and *target.y*

The result is an *nx x ny* matrix of values for *F(y|x)*. Let $f(x_j)$ represent the *ny*-vector of values of $F(y|x_j)$, and let f_k indicate the entry of $F(y|x_j)$ associated with y_k, k = 1,...,ny. Finally, let τ represent an entry of *taumat*. Then the value of *yhat.target* associated with quantile τ and x_j is the largest value of f_k such that $f_k < \tau < f_k + 1$. The resulting *nx x length(taumat)* matrix is available after estimation as *yhat.target*. The *smooth12* is used to interpolate each column of *yhat.target* to span the full vector of original values of *x*. The result is the *n x length(taumat)* matrix *yhat*.

Note: The default bandwidth may prove too small if there are regions where *x* is sparse. It may be necessary to experiment with larger bandwidths for *hx* and *hy*. The function *qreglwr* is more flexible, allowing nearest neighbor approaches as well as fixed bandwidths.

Value

yhat	Matrix of quantile predictions. Dimension is *n x length(taumat)*
yhat.target	Matrix of quantile predictions at target values of *x*. Dimension is *length(targetx) x length(taumat)*.
targetx	Vector of target values for *x*.
targety	Vector of target values for *y*.
taumat	Vector of target quantile values.
hx	Bandwidth for *x*.
hy	Bandwidth for *y*.

References

Li, Oi and Jeffrey Scott Racine. *Nonparametric Econometrics: Theory and Practice*. Princeton, NJ: Princeton University Press, 2007. Chapter 6.

See Also

condens

Examples

```
data(dupage99)
dupage99$ratio <- dupage99$av/dupage99$price
o <- order(dupage99$price)
dupage99 <- dupage99[o,]
attach(dupage99)
price <- price/1000

fit <- qregcdf(ratio~price)
ymin = min(fit$yhat)
ymax = max(fit$yhat)
plot(price, fit$yhat[,1],type="l",xlab="Sales Price (1000s)",ylab="Assessment Ratio",
  ylim=c(ymin,ymax),main="Nonparametric Conditional CDF Quantile Regression")
for (j in seq(2,5)) {
  lines(price,fit$yhat[,j])
}
fit$hx
fit$hy
```

---

qregcpar *Conditionally Parametric LWR Quantile Estimation*

Description

Estimates a model of the form *y = XB(z) + u* using locally weighted quantile regression for a set of user-provided quantiles. *z* can include one or two variables.

Usage

```
qregcpar(form,nonpar,taumat=c(.10,.25,.50,.75,.90),
  window=.25,bandwidth=0,kern="tcub",distance="Mahal",
  target=NULL,data=NULL)
```

Arguments

form	Model formula
nonpar	List of either one or two variables for z. Formats: *qregcpar(y~xlist, nonpar=~z1, ...)* or *qregcpar(y~xlist, nonpar=~z1+z2, ...).* Important: note the "~" before the first z variable.
taumat	Vector of target quantiles. Default: *taumat*=c(.10,.25,.50,.75,.90) .
window	Window size. Default: 0.25.
bandwidth	Bandwidth. Default: not used.
kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
distance	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *qregcpar* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
target	If *target* = *NULL*, uses the *maketarget* command to form targets using the values specified for *window*, *bandwidth*, and *kern*. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
data	A data frame containing the data. Default: use data in the current working directory

Details

The list of explanatory variables is specified in the base model formula while *Z* is specified using *nonpar*. *X* can include any number of explanatory variables, but *Z* must have at most two.

The estimated value of *y* at a target value z_0 and a quantile τ is the predicted value from a weighted quantile regression of y on *X* with weights given by *K*. When *Z* includes a single variable, *K* is a simple kernel weighting function: $K((z - z_0)/(sd(z) * h))$. When *Z* includes two variables (e.g, nonpar=~z1+z2), the method for specifying *K* depends on the *distance* option. Under either option, the *i*th row of the matrix *Z* = (z1, z2) is transformed such that $z_i = \sqrt{z_i * V * t(z_i)}$. Under the "Mahal" option, *V* is the inverse of cov(*Z*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*Z*)). After this transformation, the weights again reduce to the simple kernel weighting function $K((z - z_0)/(sd(z) * h))$.

The great circle formula is used to define *K* when *distance* = *"Latlong"*; in this case, the variable list for *nonpar* must be listed as *nonpar* = *~latitude+longitude* (or *~lo+la* or *~lat+long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude

are listed does not matter and the function only looks for the first two letters to determine which variable is latitude and which is the longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance = "Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option. *h* is specified by the *bandwidth* or *window* option.

For each quantile, the estimated coefficient matrix, *xcoef*, includes an intercept (the first column in *k* of *xcoef*) and the coefficients for the explanatory variables. The dimension of *xcoef* is *n x ntau x k*.

Estimation can be very slow when *target = "alldata"*. The *maketarget* command can be used to identify target points.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|)I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target` The target points for the original estimation of the function.

`xcoef.target` The matrix of estimated coefficients, *B(z)*, at the target values of *z*. Dimension = *ntarget x ntau x k*, where *ntarget* = number of target poitns, *ntau* = number of quantiles, and *k* = number of explanatory variables including the intercept.

`xcoef.target.se` The matrix of standard errors for *B(z)* at the target values of *z*. Dimension = *ntarget x ntau x k.*

`xcoef` The matrix of estimated coefficients, *B(z)*, at the original data points. Dimension = *n x ntau x k.*

`xcoef.se` The matrix of standard errors for *B(z)* with *z* evaluated at all points in the data set. Dimension = *n x ntau x k.*

`yhat` The matrix of predicted values of *y* at the original data points. Dimension = *n x ntau.*

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Loader, Clive. *Local Regression and Likelihood.* New York: Springer, 1999.

Koenker, Roger. *Quantile Regression.* New York: Cambridge University Press, 2005. Chapter 7 and Appendix A.9.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

McMillen, Daniel P. and Christian Redfearn, "Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions," *Journal of Regional Science* 50 (2010), 712-733.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics.* New York: Cambridge University Press, 1999.

See Also

qreglwr

Examples

```
data(cookdata)
cookdata$obs <- seq(1,nrow(cookdata))
cookdata <- cookdata[!is.na(cookdata$FAR),]
par(ask=TRUE)

# 1. CPAR LWR estimates, y = a(DCBD) + b(dcbd)*DCBD + u
fit <- qregcpar(LNFAR~DCBD,nonpar=~DCBD, taumat=c(.10,.50,.90),
  kern="bisq", window=.30, data=cookdata)
o <- order(cookdata$DCBD)
plot(cookdata$DCBD[o], fit$yhat[o,1],type="l", main="Log Floor Area Ratio",
  xlab="Distance from CBD",ylab="Log FAR")
lines(cookdata$DCBD[o], fit$yhat[o,2])
lines(cookdata$DCBD[o], fit$yhat[o,3])

## Not run:
# 2. CPAR estimates, y = a(lat,long) + b(lat,long)xDCBD + u
fit <- qregcpar(LNFAR~DCBD, nonpar=~LATITUDE+LONGITUDE, taumat=c(.10,.90),
  kern="bisq", window=.30, distance="LATLONG", data=cookdata)
plot(cookdata$DCBD, cookdata$LNFAR,main="Log Floor Area Ratio",
  xlab="Distance from CBD",ylab="Log FAR")
points(cookdata$DCBD, fit$yhat[,1], col="red")
plot(cookdata$DCBD, cookdata$LNFAR,main="Log Floor Area Ratio",
  xlab="Distance from CBD",ylab="Log FAR")
points(cookdata$DCBD, fit$yhat[,2], col="red")

library(RColorBrewer)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap$yhat10[cookdata$obs] <- fit$yhat[,1]
cmap$yhat90[cookdata$obs] <- fit$yhat[,2]
cmap$yhat1090 <- cmap$yhat90 - cmap$yhat10
brks <- seq(min(cmap$yhat1090,na.rm=TRUE),max(cmap$yhat1090,na.rm=TRUE),length=9)
spplot(cmap,"yhat1090",at=brks,col.regions=rev(brewer.pal(9,"RdBu")),
  main="Difference between .10 and.90 Quantiles")

## End(Not run)
```

qreglwr *Locally Weighted Quantile Regression*

Description

Estimates a model of the form *y = f(x)* using locally weighted quantile regression for a set of user-provided quantiles. *x* can include either one or two variables. Returns estimated values, derivatives, and standard errors for both *f(x)* and *df(x)/dx*.

Usage

```
qreglwr(form,taumat=c(.10,.25,.50,.75,.90), window=.25,bandwidth=0,
  kern="tcub", distance="Mahal",target=NULL,data=NULL)
```

Arguments

form	Model formula
taumat	Vector of target quantiles. Default: *taumat*=c(.10,.25,.50,.75,.90) .
window	Window size. Default: 0.25.
bandwidth	Bandwidth. Default: not used.
kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
distance	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *qreglwr* looks for the first two letters to determine which variable is latitude and which is longitude, so the data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
target	If *target = NULL*, uses the *maketarget* command to form targets using the values specified for *window*, *bandwidth*, and *kern*. If *target="alldata"*, each observation is used as a target value for *x*. A set of target values can be supplied directly.
data	A data frame containing the data. Default: use data in the current working directory.

Details

Serves as an interface to the *quantreg* package. Uses a kernel weight function in *quantreg's* "weight" option to estimate quantile regressions at a series of target values of *x*. *x* may include either one or two variables. The target values are found using *locfit's* adaptive decision tree approach. The predictions are then interpolated to the full set of *x* values using the *smooth12* command. If *alldata=T*, the procedure is applied to every value of *x* rather than a set of target points.

The weights at a target value x_0 are given by $K(\psi/h)$, where ψ is a measure of the distance between *x* and x_0 and *h* is the bandwidth or window. When *x* includes a single variable, $\psi = x - x_0$. When *x* includes two variables, the method for specifying ψ depends on the *distance* option. If

distance="Mahal" or *distance="Euclid"*, the *i*th row of the matrix *X* = (x1, x2) is transformed such that $x_i = sqrt(x_i * V * t(x_i))$. Under the "Mahal" option, *V* is the inverse of cov(*X*). Under the *"Euclid"* option, *V* is the inverse of diag(cov(*X*)). By reducing x from two dimensions to one, this transformation leads again to the simple kernel weighting function $K((x - x_0)/(sd(x) * h))$. *h* is specified by the bandwidth or window options.

The great circle formula is used to define *K* when *distance = "Latlong"*; in this case, the explanatory variable list must be specified as *~latitude+longitude* (or *~lo+la* or *~lat+long*, etc), with the longitude and latitude variables expressed in degrees (e.g., -87.627800 and 41.881998 for one observation of longitude and latitude, respectively). The order in which latitude and longitude are listed does not matter and the function only looks for the first two letters to determine which variable is latitude and which is longitude. It is important to note that the great circle distance measure is left in miles rather than being standardized. Thus, the window option should be specified when *distance = "Latlong"* or the bandwidth should be adjusted to account for the scale. The kernel weighting function becomes *K(distance/h)* under the *"Latlong"* option.

Since *qreglwr* estimates weighted quantile regressions of the dependent variable, *y*, on $x - x_0$, the intercept provides an estimate of *y* at x_0 and β provides an estimate of the slope of the quantile line, *dy/dx*, at x_0. *quantreg's* standard error for the intercept is stored in *ytarget.se* (target points) and *yhat.se* (all observations). The standard errors for the slopes are stored in *dtarget1.se, dtarget2.se, dhat1.se,* and *dhat2.se*.

When *alldata=T*, each data point in turn is used as a target point, x_0. Fixed bandwidths may prove too small if there are regions where *x* is sparse. A nearest neighbor approach is generally preferable (e.g, window=.50). Estimation can be very slow when *target = "alldata"*. The *maketarget* command can be used to identify target points. The *smooth12* command is then used to interpolate the coefficient estimates and standard errors.

Available kernel weighting functions include the following:

Kernel	Call abbreviation	Kernel function K(z)
Rectangular	"rect"	$\frac{1}{2}I(\|z\| < 1)$
Triangular	"tria"	$(1 - \|z\|)I(\|z\| < 1)$
Epanechnikov	"epan"	$\frac{3}{4}(1 - z^2) * I(\|z\| < 1)$
Bi-Square	"bisq"	$\frac{15}{16}(1 - z^2)^2 * I(\|z\| < 1)$
Tri-Cube	"tcub"	$\frac{70}{81}(1 - \|z\|^3)^3 * I(\|z\| < 1)$
Tri-Weight	"trwt"	$\frac{35}{32}(1 - z^2)^3 * I(\|z\| < 1)$
Gaussian	"gauss"	$(2\pi)^{-.5}e^{-z^2/2}$

Value

`target`	The target points for the original estimation of the function.
`ytarget`	The matrix of predicted values of *y* at the target points, by quantile. Rows represent targets; columns are quantiles.
`dtarget1`	The matrix of estimated derivatives *dy/dx1* at the target points, by quantile. Rows represent targets; columns are quantiles.
`dtarget2`	The matrix of estimated derivatives *dy/dx2* at the target points, by quantile. Rows represent targets; columns are quantiles. All zeros if the model has only one explanatory variable.

`ytarget.se` The matrix of standard errors for the predicted values of *y* at the target points, by quantile. Rows represent targets; columns are quantiles.

`dtarget1.se` The matrix of standard errors for the derivatives *dy/dx1* at the target points, by quantile. Rows represent targets; columns are quantiles.

`dtarget2.se` The matrix of standard errors for the derivatives *dy/dx2* at the target points, by quantile. Rows represent targets; columns are quantiles. All zeros if the model has only one explanatory variable.

`yhat` The matrix of predicted values of *y* for the full data set, by quantile. Dimension = n x length(taumat).

`dhat1` The matrix of estimated derivatives *dy/dx1* for the full data set, by quantile. Dimension = n x length(taumat).

`dhat2` The matrix of estimated derivatives *dy/dx2* for the full data set, by quantile. Dimension = n x length(taumat). All zeros if the model has only one explanatory variable.

`yhat.se` The matrix of standard errors for the predicted values of *y* for the full data set, by quantile. Dimension = n x length(taumat).

`dhat1.se` The matrix of standard errors for the estimated derivatives *dy/dx1* for the full data set, by quantile. Dimension = n x length(taumat).

`dhat2.se` The matrix of standard errors for the estimated derivatives *dy/dx2* for the full data set, by quantile. Dimension = n x length(taumat). All zeros if the model has only one explanatory variable.

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Koenker, Roger. *Quantile Regression.* New York: Cambridge University Press, 2005. Chapter 7 and Appendix A.9.

Loader, Clive. *Local Regression and Likelihood*. New York: Springer, 1999.

See Also

`lwr`

Examples

```
data(cookdata)
cookdata <- cookdata[cookdata$CHICAGO==1,]
cookdata$obs <- seq(1:nrow(cookdata))
cookdata <- cookdata[cookdata$CHICAGO==1&cookdata$POPULATION>0,]
par(ask=TRUE)

# lndens = f(dcbd)
fit <- lwr(LNDENS~DCBD,window=.20,data=cookdata)
fit1 <- qreglwr(LNDENS~DCBD,taumat=c(.10,.50,.90),window=.30,kern="rect",data=cookdata)
o <- order(cookdata$DCBD)
```

```
ymin = min(fit1$yhat)
ymax = max(fit1$yhat)
plot(cookdata$DCBD[o], fit$yhat[o], type="l", ylim=c(ymin,ymax),
  xlab="Distance to CBD", ylab="Log of Population Density")
lines(cookdata$DCBD[o], fit1$yhat[o,1], col="red", lty="dashed")
lines(cookdata$DCBD[o], fit1$yhat[o,2], col="red")
lines(cookdata$DCBD[o], fit1$yhat[o,3], col="red", lty="dashed")
legend("topright", c("LWR", "tau = 50", "tau = 10, 90"), col=c("black","red", "red"),
  lwd=1, lty=c("solid","solid","dashed"))

## Not run:
library(RColorBrewer)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1,]
# lndens = f(longitude, latitude), weights are function of straight-line distance
fit <- qreglwr(LNDENS~LONGITUDE+LATITUDE,taumat=c(.10,.50,.90),window=.20,data=cookdata)
cmap$lwr10[cookdata$obs] <- fit$yhat[,1]
cmap$lwr50[cookdata$obs] <- fit$yhat[,2]
cmap$lwr90[cookdata$obs] <- fit$yhat[,3]
cmap$lwr1090[cookdata$obs] <- fit$yhat[,3] - fit$yhat[,1]
brks <- seq(min(cmap$lwr10,na.rm=TRUE),max(cmap$lwr10,na.rm=TRUE),length=9)
spplot(cmap,"lwr10",at=brks,col.regions=rev(brewer.pal(8,"RdBu")),
   main="Log Density Estimates, tau = .10")
brks <- seq(min(cmap$lwr50,na.rm=TRUE),max(cmap$lwr50,na.rm=TRUE),length=9)
spplot(cmap,"lwr50",at=brks,col.regions=rev(brewer.pal(8,"RdBu")),
   main="Log Density Estimates, tau = .50")
brks <- seq(min(cmap$lwr90,na.rm=TRUE),max(cmap$lwr90,na.rm=TRUE),length=9)
spplot(cmap,"lwr90",at=brks,col.regions=rev(brewer.pal(8,"RdBu")),
   main="Log Density Estimates, tau = .90")
brks <- seq(min(cmap$lwr1090,na.rm=TRUE),max(cmap$lwr1090,na.rm=TRUE),length=9)
spplot(cmap,"lwr1090",at=brks,col.regions=rev(brewer.pal(8,"RdBu")),
   main="Difference in Log Density, tau = .90 - .10")

## End(Not run)
```

---

qregsim1 *Changes in Distributions Implied by Quantile Regression Estimates*

Description

Uses quantile regression results to simulate the effects of an explanatory variable on the distribution of the dependent variable.

Usage

```
qregsim1(formall, formx, bmat, taumat, xvalues=NULL, ytarget=NULL,
  xcolors=NULL,graphx=TRUE, graphy=TRUE, graphsim=TRUE, histogram=FALSE,
  histfreq=FALSE, yname=NULL, xname=NULL, nsim=0, bwadjust=1,
  legloc="topright",data=NULL)
```

Arguments

formall	Formula with the dependent variable and *all* explanatory variables, as provided to the *qregbmat* or *qregcpar* command.
formx	The explanatory variable to be analyzed. Specified as a formula, e.g., formx=~x.
bmat	Matrix of coefficient estimates from the *qregbmat* or *qregcpar* command.
taumat	The vector of quantile values represented in *bmat*.
xvalues	Vector of explanatory variable values for discrete changes. If *xvalues = NULL*, the values are set to *xvalues <- quantile(x,c(.25,.75))*.
ytarget	Vector of target values for the density functions involving y. The values in *ytarget* are used as the target points for the predicted and actual values of y. The same values of *ytarget* are also used as the target values for the simulations.
xcolors	Vector of colors for the density function graphs. Default for two values of *xvalues* is c("black","red"). For more than two values, the default is drawn from the *RColorBrewer* package: *xcolors = brewer.pal(nx,"Blues")*.
graphx	If *graphx=T*, presents the kernel density function for the explanatory variable.
graphy	If *graphy=T*, presents density functions for the actual and predicted values of the dependent variable.
graphsim	If *graphsim=T*, presents graphs of the density functions for the predicted values of y at the values specified in *xvalues*.
histogram	If *histogram=T* and *graphx=T*, the density function for the explanatory variable is presented as a histogram. Not relevant if *graphx=F*.
histfreq	If *histogram=T* and *graphx=T*, the histogram is presented using frequencies rather than densities. Not relevant if *graphx=F* or *histogram=F*.
yname	A label used for the dependent variable in the density graphs, e.g., *yname = "Log of Sale Price"*.
xname	A label for graphs involving the explanatory variable, e.g., *xname = "x1"*.
nsim	Number of simulations for quantile distributions. Default: *nsim=0*.
bwadjust	Factor used to adjust bandwidths for kernel densities. Smoother functions are produced when *bwadjust*>1. Passed directly to the *density* function's *adjust* option. Default: *bwadjust=1*.
legloc	The legend location.
data	A data frame containing the data. Default: use data in the current working directory.

Details

The conditional quantile function is $y(\tau) = \alpha(\tau) + \beta(\tau) * x + \lambda(\tau) * z$. The complete model specification is listed in *formall*, while x is specified in *formx*, e.g., *formall <- y~x+z* and *formx <- ~x*. When *nsim = 0*, the *qregsim1* command simply calculates predicted values of y at each value of x listed in *xvalues* and at each τ list in *taumat*. Thus, the first column of *densyhat* holds the estimated density function for $\hat{y}(\tau) = \hat{\alpha}(\tau) + \hat{\beta}(\tau) * xvalues[1] + \hat{\lambda}(\tau) * z$, the second column holds the predictions at *xvalues[2]*, and so on. The estimates are evaluated at each value of τ, which

leads to an *n x length(taumat)* set of predictions for each value of *xvalues*. Kernel density estimates are then calculated for these predictions.

The *qregsim1* command can follow either *qregbmat* or *qregcpar*. All that differs is the dimension of *bmat*: using *qregcpar*, the coefficients vary by observation.

If *nsim > 0*, the *qregsim1* function uses a simulation procedure based on the Machado-Mata (2005) approach to simulate the effect of an explanatory variable on the distribution of the dependent variable. The function begins by drawing (with replacement) *nsim* values from the rows of the explanatory variable matrix and *nsim* values of τ. With *nsim* values of both the explanatory variables and coefficient vectors, the predictions are simply $\hat{y}_j(\tau) = \hat{\alpha}(\tau) + \hat{\beta}(\tau) * xvalues[j] + \hat{\lambda}(\tau) * z$ for $j = 1, ..., length(xvalues)$.

The *n x length(xvalues) matrix* densyhat holds the full set of predictions.

In addition to the predicted values of y at *xvalues*, the *qregsim1* command can produce the following:

1. The estimated density function for the explanatory variable (if *graphx=T*). Presented as a histogram if *histogram=T*.

2. The estimated quantile regression coefficients for the variable specified by *formx*. This graph is produced if *graphb=T*.

Value

`ytarget`	The values for the horizontal axis of the quantile predictions at *xvalues*.
`densyhat`	Matrix of predictions for the predicted values of y at the values specified in *xvalues*. The dimension of the matrix is *n x length(xvalues)*.
`densy1`	The values of the density function for the actual values of the dependent variable. *densy1=NULL* if *graphy=F*.
`densy2`	The values of the density function for the quantile regression predictions of the dependent variable. *densy2=NULL* if *graphy=F*.

References

Koenker, Roger. *Quantile Regression*. New York: Cambridge University Press, 2005.

Machado, J.A.F. and Mata, J., "Counterfactual Decomposition of Changes in Wage Distributions using Quantile Regression," *Journal of Applied Econometrics* 20 (2005), 445-465.

McMillen, Daniel P., "Changes in the Distribution of House Prices over Time: Structural Characteristics, Neighborhood or Coefficients?" *Journal of Urban Economics* 64 (2008), 573-589.

See Also

`qregbmat`

`qregsim2`

`qregcpar`

Examples

```
par(ask=TRUE)
data(matchdata)
matchdata$age <- matchdata$year - matchdata$yrbuilt
tvect <- seq(.10,.90,.10)
bmat <- qregbmat(lnprice~lnland+lnbldg+age+factor(year), data=matchdata,
  graph.factor=TRUE,taumat=tvect,graphb=FALSE)

xvect <- signif(quantile(matchdata$lnland,c(.25,.75)),4)
fit <- qregsim1(lnprice~lnland+lnbldg+age+factor(year),~lnland, bmat,
  tvect,xvalues=xvect,data=matchdata)
#Simulation for 1995
smalldata <- matchdata[matchdata$year==1995,]
bmat95 <- bmat[,1:4]
fit <- qregsim1(lnprice~lnland+lnbldg+age,~lnland,bmat95,tvect,
  xvalues=xvect, data=smalldata)
#Simulation for 2005
smalldata <- matchdata[matchdata$year==2005,]
bmat05 <- bmat[,1:4]
bmat05[,1] <- bmat05[,1] + bmat[,5]
fit <- qregsim1(lnprice~lnland+lnbldg+age,~lnland,bmat05,tvect,
  xvalues=xvect, data=smalldata)
```

---

qregsim2 *Machado-Mata Decomposition of Changes in Distributions*

Description

Decomposes quantile regression estimates of changes in the distribution of a dependent variable into the components associated with changes in the distribution of the explanatory variables and the coefficient estimates.

Usage

```
qregsim2(formall, formx, dataframe1, dataframe2, bmat1, bmat2,
  graphx=TRUE, graphb=TRUE, graphy=TRUE, graphdy=TRUE, nbarplot=10,
  yname=NULL, xnames=NULL, timenames=c("1","2"),
  leglocx="topright",leglocy="topright",leglocdy="topright",
  nsim=20000, bwadjx=1,bwadjy=1,bwadjdy=1)
```

Arguments

formall	Model formula. Must match the model formula used for *qregbmat*.

formx	Model formula for the variables used for the decompositions, e.g., *formx=~x1+x2*. The coefficients and variables for the other variables are held at their time 2 values for the simulations.
dataframe1	The data frame for regime 1. Should include all the variables listed in *formall*.
dataframe2	The data frame for regime 2. Should include all the variables listed in *formall*.
bmat1	Matrix of values for regime 1 quantile coefficient matrices; the output from running *qregbmat* using *dataframe1*.
bmat2	Matrix of values for regime 2 quantile coefficient matrices; the output from running *qregbmat* using *dataframe2*.
graphx	If *graphx=T*, presents kernel density estimates of each of the explanatory variables in *formx*.
graphb	If *graphb=T*, presents graphs of the quantile coefficient estimates for the variables in *formx*.
graphy	If *graphy=T*, presents of the predicted values of *y* for time1, time2, and the counterfactual.
graphdy	If *graphdy=T*, presents graphs of the changes in densities.
nbarplot	Specifies the maximum number of values taken by an explanatory variable before bar plots are replaced by smooth kernel density functions. Only relevant when *graphx = T*.
yname	A label used for the dependent variable in the density graphs, e.g., *yname = "Log of Sale Price"*.
xnames	Labels for graphs involving the explanatory variables, e.g., *xnames = "x1"* for one explanatory variable, or *xnames = c("x1","x2")* for two variables.
timenames	A vector with labels for the two regimes. Must be entered as a vector with character values. Default: c("1","2").
leglocx	Legend location for density plots of the explanatory variables, e.g., *leglocx = "topright"* for one explanatory variable, or *leglocx = c("topright","topleft")* for two variables.
leglocy	Legend location for density plots of predicted values of the dependent variable. Default: *leglocy = "topright"*.
leglocdy	Legend location for plot of density changes. Default: *leglocdy = "topright"*.
nsim	Number of simulations for the decompositions.
bwadjx	Factor used to adjust bandwidths for kernel density plots of the explanatory variables. Smoother functions are produced when *bwadjust*>1. Passed directly to the *density* function's *adjust* option. Default: *bwadjx=1*.
bwadjy	Factor used to adjust bandwidths for kernel density plots predicted values of the dependent variable.
bwadjdy	Factor used to adjust bandwidths for plots of the kernel density changes.

Details

The base models are $y_1 = X_1\beta_1 + Z_1\gamma_1$ for regime 1 and $y_2 = X_2\beta_2 + Z_2\gamma_2$ for regime 2. The counterfactual model is $y_{12} = X_1\beta_2 + Z_2\gamma_2$. The full list of variable (both X and Z) are provided by *form*; this list must correspond exactly with the list provided to *qregbmat*. The subset of variables that are the subject of the decompositions are listed in *formx*.

The matrices *bmat1* and *bmat2* are intended to represent the output from *qregbmat*. The models must include the same set of explanatory variables, and the variables must be in the same order in both *bmat1* and *bmat2*. In contrast, the data frames *dataframe1* and *dataframe2* can have different numbers of observations and different sets of explanatory, as long as they include the dependent variable and the variables listed in *bmat1* and *bmat2*.

The output from *qregsim2* is a series of graphs. If all options are specified, the graphs appear in the following order:

1. Kernel density estimates for each variable listed in *formx*. Estimated using *density* with default bandwidths and the specified value for *bwadjx*. Not shown if *graphx=F*. The *xnames* and *leglocx* options can be used to vary the names used to label the x-axis and the legend location.

2. Quantile coefficient estimates for the variables listed in *formx*. Not listed if *graphb=F*.

3. Kernel density estimates for the predicted values of $X_1\beta_1 + Z_1\gamma_1$ and $X_2\beta_2 + Z_2\gamma_2$, and the counterfactual, $X_1\beta_2 + Z_2\gamma_2$. Estimated using *density* with default bandwidths and the specified value for *bwadjy*. Not shown if *graphy=F*. The label for the x-axis can be varied with the *yname* option. The three estimated density functions are returned after estimation as *yhat11*, *yhat22*, and *yhat12*.

4. A graph showing the change in densities, d2211 = f22 - f11, along with the Machado-Mata decomposition showing:

(a) the change in densities due to the variables listed in *formx*: d2212 = f22 - f12.

(b) the change in densities due to the coefficients: d1211 = f12 - f11.

These estimates are returned after estimation as *d2211*, *dd2212*, and *d1211*. The density changes are not shown if *graphdy=F*. The label for the x-axis can be varied with the *yname* option. The bandwidth for the original density functions f11, f22, and f12 can be varied with *bwadjdy*. It is generally desirable to set *bwadjdy* > *bwadjy* because additional smoothing is needed to make the change in densities appear smooth.

The distributions are simulated by drawing *nsim* samples with replacement from *xobs1 <- seq(1:n1)*, *xobs2 <- seq(1:n2)*, and *bobs <- seq(1:length(taumat))*. The commands for the simulations are:

```
xobs1 <- sample(seq(1:n1),nsim,replace=TRUE)

xobs2 <- sample(seq(1:n2),nsim,replace=TRUE)

bobs <- sample(seq(1:ntau),nsim,replace=TRUE)

xhat1 <- allmat1[xobs1,]

xhat2 <- allmat2[xobs2,]

znames <- setdiff(colnames(allmat1),colnames(xmat1))

if (identical(znames,"(Intercept)")) xhat12 <- xhat1

if (!identical(znames,"(Intercept)"))

xhat12 <- cbind(xhat2[,znames],xhat1[,colnames(xmat1)])

xhat12 <- xhat12[,colnames(allmat1)]
```

bhat1 <- bmat1[bobs,]

bhat2 <- bmat2[bobs,]

where *allmat* and *xmat* denote the matrices defined by explanatory variables listed in *formall* (including the intercet) and *formx*. Since the bandwidths are simply the defaults from the *density* function, they are likely to be different across regimes as the number of observations and the standard deviations may vary across times. Thus, the densities are re-estimated using the average across regimes of the original bandwidths.

Value

`ytarget`	The values for the x-axis for the density functions.
`yhat11`	The kernel density function for $X_1\beta_1 + Z_1\gamma_1$.
`yhat22`	The kernel density function for $X_2\beta_2 + Z_2\gamma_2$.
`yhat12`	The kernel density function for $X_1\beta_2 + Z_2\gamma_2$.
`d2211`	The difference between the density functions for $X_2\beta_2+Z_2\gamma_2$ and $X_1\beta_1+Z_1\gamma_1$. Will differ from yhat22 - yhat11 if *bwadjy* and *bwadjdy* are different.
`d2212`	The difference between the density functions for $X_2\beta_2+Z_2\gamma_2$ and $X_1\beta_2+Z_2\gamma_2$. Will differ from yhat22 - yhat12 if *bwadjy* and *bwadjdy* are different.
`d1211`	The difference between the density functions for $X_1\beta_2+Z_2\gamma_2$ and $X_1\beta_1+Z_1\gamma_1$. Will differ from yhat12 - yhat11 if *bwadjy* and *bwadjdy* are different.

References

Koenker, Roger. *Quantile Regression*. New York: Cambridge University Press, 2005.

Machado, J.A.F. and Mata, J., "Counterfactual Decomposition of Changes in Wage Distributions using Quantile Regression," *Journal of Applied Econometrics* 20 (2005), 445-465.

McMillen, Daniel P., "Changes in the Distribution of House Prices over Time: Structural Characteristics, Neighborhood or Coefficients?" *Journal of Urban Economics* 64 (2008), 573-589.

See Also

dfldens

qregbmat

qregsim1

qregcpar

qreglwr

Examples

```
par(ask=TRUE)

n = 5000
set.seed(484913)
x1 <- rnorm(n,0,1)
u1 <- rnorm(n,0,.5)
```

```
y1 <- x1 + u1

# no change in x.  Coefficients show quantile effects
tau <- runif(n,0,.5)
x2 <- x1
y2 <- (1 + (tau-.5))*x2 + .5*qnorm(tau)

dat <- data.frame(rbind(cbind(y1,x1,1), cbind(y2,x2,2)))
names(dat) <- c("y","x","year")
bmat1 <- qregbmat(y~x,data=dat[dat$year==1,],graphb=FALSE)
bmat2 <- qregbmat(y~x,data=dat[dat$year==2,],graphb=FALSE)
fit1 <- qregsim2(y~x,~x,dat[dat$year==1,],dat[dat$year==2,],
  bmat1,bmat2,bwadjdy=2)

# Distribution of x changes.  Coefficients and u stay the same
x2 <- rnorm(n,0,2)
y2 <- x2 + u1
dat <- data.frame(rbind(cbind(y1,x1,1), cbind(y2,x2,2)))
names(dat) <- c("y","x","year")
bmat1 <- qregbmat(y~x,data=dat[dat$year==1,],graphb=FALSE)
bmat2 <- qregbmat(y~x,data=dat[dat$year==2,],graphb=FALSE)
fit1 <- qregsim2(y~x,~x,dat[dat$year==1,],dat[dat$year==2,],
  bmat1,bmat2,bwadjdy=2)
```

`qregspiv` *IV Estimator for the Spatial AR Quantile Model*

Description

Uses the Kim and Muller (2004) or Chernozhukov and Hansen (2006) method to estimate a quantile version of the spatial AR Model

Usage

```
qregspiv(form,wy=NULL,wmat=NULL,inst=NULL,winst=NULL,shpfile=NULL,
  tau=.5,rhomat=NULL,printsariv=FALSE,silent=FALSE,
  nboot=100,alpha=.05,data=NULL)
```

Arguments

`form`	Model formula
`wy`	The *WY* variable. Default: not specified; program attempts to calculate *WY* using *wmat* or *shpfile*.
`wmat`	Directly enter *wmat* rather than creating it from a shape file. Default: not specified. *wmat* is needed unless *WY* is provided and a full instrument list is specified using *inst*. Default: *W* = NULL.

inst	List of instruments *not* to be pre-multiplied by *W*. Entered as *inst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
winst	List of instruments to be pre-multiplied by *W* before use. Entered as *winst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
shpfile	Shape file used to construct *W* based on first order contiguity using a queen criterion. Needed if *wmat* is not provided when the program requires it.
tau	The quantile. Default: *tau* = .5
rhomat	A vector of values for ρ. If *rhomat*=NULL, uses the Kim and Muller (2004) two-stage approach to estimate the model. If *rhomat* is a vector with two or more entries, uses the Chernozhukov and Hansen (2006) IV approach to estimate the model. Default: *rhomat=NULL*.
printsariv	If *TRUE*, also estimates a standard spatial AR model using an IV approach. Instruments for WY are based on the *inst* and *winst* options. Default: *printsariv=FALSE*.
silent	If *TRUE*, no results are printed. Useful for Monte Carlo.
nboot	The number of simulations for the bootstrap standard errors. Needed for the Kim and Muller (2004) model. Default: *nboot*=100.
alpha	Probability for the confidence intervals, calculated by the percentile method for the Kim and Muller (2004) model. Default: *alpha*=.05, i.e., a 95 percent confidence interval.
data	A data frame containing the data. Default: use data in the current working directory.

Details

The procedure is intended for quantile estimation of the spatial AR model, $Y = \rho WY + X\beta + u$. It can also be for quantile IV estimation of any model with one endogenous explanatory variable.

Kim and Muller (2004):

The Kim and Muller 2004 estimation procedure is the default. The procedure has two stages. In the first stage, an instrumental variable is constructed for *WY* using the predicted values from a quantile regression of *WY* on a set of instruments, *Z*. The second stage is a quantile regression of *Y* on *X* and the predicted values of *WY*. The same quantile, *tau*, is used for both regressions.

Standard errors are calculated using a simple bootstrap estimator. New samples are constructed by drawing with replacement from the rows of the data frame holding *y*, *WY*, *X*, and *Z*. Both stages are re-estimated *nboot* times using the series of bootstrap samples. The bootstrap standard errors are the standard deviations of the *nboot* re-calculations of the coefficient estimates. The confidence intervals are based on the percentile method: for any coefficient *b*, the 1-*alpha* confidence interval is (*quantile(b, alpha/2)*, *quantile(b, 1-alpha/2)*).

Chernozhukov and Hansen (2006):

The Chernozhukov and Hansen (2006) procedure is used when a vector of possible values for ρ is specified using the *rhomat* option, e.g., *rhomat* = seq(0,.9,.05). The *qregspiv* command implements a simple version of the Chernzhukov and Hansen estimator in which the explanatory variable *WY* is replaced by the predicted values from an OLS regression of *WY* on the instruments, *Z*. This instrumental variable is then used as an explanatory variable for a series of quantile regressions of $Y - \rho WY$ on *X* and $\hat{WY}$ – one regression for each value of ρ listed in *rhomat*. The estimated

value of ρ is the value that leads the coefficient on $\hat{WY}$ to be closest to zero. After finding $\hat{\rho}$, the estimated values of β are calculated by a quantile regression of $Y - \hat{\rho}WY$ on *X*.

Standard errors are based on equations 3.13 and 3.14 in Chernozhukov and Hansen (2006). Let *e* represent the residuals from the quantile regression of $Y - \hat{\rho}WY$ on *X*, and define $f_i = I(|e_i| < h)/(2h)$, where $h = 1.06 * sd(e) * n^{-.2}$. Also, let Φ represent the predicted value of *WY* from an OLS regression of *WY* on *Z*, and let *D* represent the actual values of *WY*. Finally, define $\Phi_i^* = f_i\Phi_i$ and $X_i^* = f_i X_i$. Then the covariance matrix for $\hat{\theta} = (\hat{\rho}, \hat{\beta})$ is .

$$V(\hat{\theta}) = J(\tau)^{-1}S(\tau)(J^t)^{-1}$$

where $J(\tau) = \begin{pmatrix} \Phi^{*t}D & \Phi^{*t}X \\ X^{*t}D & X^{*t}X \end{pmatrix}$ and $S(\tau) = \tau(1-\tau)\begin{pmatrix} \Phi^{t}D & \Phi^{t}X \\ X^{t}D & X^{t}X \end{pmatrix}$

Instruments:

By default, the instrument list includes *X* and *WX*, where *X* is the original explanatory variable list and *W* is the spatial weight matrix. It is also possible to directly specify the full instrument list or to include only a subset of the *X* variables in the list that is to be pre-multiplied by *W*. The results of both the quantile IV estimator and the standard IV estimator can be quite sensitive to the choice of instruments for the spatial AR model.

Let *list1* and *list2* be user-provided lists of the form *list=~z1+z2*. The combinations of defaults (*NULL*) and lists for *inst* alter the final list of instruments as follows:

inst = NULL, *winst = NULL*: *Z = (X, WX)*
inst = list1, *winst = NULL*: *Z = list1*
inst = NULL, *winst = list2*: *Z = (X, W*list2)*
inst = list1, *winst = list2*: *Z = (list1, W*list2)*

Note that when *inst=list1* and *winst=NULL* it is up to the user to specify at least one variable in *list1* that is not also included in *X*.

Non-Quantile IV Estimates:

Standard, non-quantile IV estimates are presented if *printsariv = T*. The first stage is a regression of *WY* on *Z*. The second stage is a regression of *Y* on *X* and the predicted values of *WY*. Let $\hat{G}$ be the matrix of explanatory variables in the second stage (i.e., $\hat{G} = (Z, \hat{WY})$), the covariance matrix is $\hat{\sigma}^2(\hat{G}'\hat{G})^{-1}$, where $\hat{\sigma}^2 = e'e/n$ and $e = Y - \hat{\rho}WY - X\hat{\beta}$. Note that the variance calculation uses actual values of *WY* while $(\hat{G}'\hat{G})^{-1}$ uses predicted values.

Value

A table showing the coefficient estimates, standard errors, and z-values. Also includes a 1-*alpha* confidence interval based on the percentile method when the Kim and Muller is estimated, i.e., when *rhomat = NULL*.

References

Chernozhukov, Victor and Christian Hansen, "Instrumental Quantile Regression Inference for Structural and Treatment Effect Models," *Journal of Econometrics* 132 (2006), 491-525.

Kim, Tae-Hwan and Christophe Muller, "Two-Stage Quantile Regression when the First Stage is Based on Quantile Regression, *Econometrics Journal* 7 (2004), 218-231.

Koenker, Roger. *Quantile Regression*. New York: Cambridge University Press, 2005.

Kostov, Philip, "A Spatial Quantile Regression Hedonic Model of Agricultural Land Prices," *Spatial Economic Analysis* 4 (2009), 53-72.

Zietz, Joachim, Emily Norman Zietz, and G. Stacy Sirmans, "Determinants of House Prices: A Quantile Regression Approach," *Journal of Real Estate Finance and Economics* 37 (2008), 317-333.

See Also

sarml

qregbmat

qregsim1

qregsim2

qregcpar

qreglwr

Examples

```
data(matchdata)
set.seed(4849189)
mdata <- matchdata[matchdata$year==2005,]
obs <- sample(seq(1,nrow(mdata)),400)
mdata <- mdata[obs,]
mdata$age <- 2005 - mdata$yrbuilt
lmat <- cbind(mdata$longitude,mdata$latitude)

fit <- makew(coormat=lmat,method="ring",ringdist=.50)
wmat <- fit$wmat
form <- lnprice~lnland+lnbldg
fit <- qregspiv(form,wmat=wmat,data=mdata,tau=.5)
```

---

repsale *Repeat Sales Estimation*

Description

Standard and Weighted Least Squares Repeat Sales Estimation

Usage

```
repsale(price0,time0,price1,time1,mergefirst=1,
 graph=TRUE,graph.conf=TRUE,conf=.95,
  stage3=FALSE,stage3_xlist=~timesale,print=TRUE)
```

Arguments

price0	Earlier price in repeat sales pair
time0	Earlier time in repeat sales pair
price1	Later price in repeat sales pair
time1	Later time in repeat sales pair
mergefirst	Number of initial periods with coefficients constrained to zero. Default: *mergefirst=1*
graph	If TRUE, graph results. Default: *graph=T*
graph.conf	If TRUE, add confidence intervals to graph. Default: *graph.conf=T*
conf	Confidence level for intervals. Default: .95
stage3	If *stage3 = NULL*, no corrections for heteroskedasticty. If *stage3="abs"*, uses the absolute value of the first-stage residuals as the dependent variable in the second-stage regression. If *stage3="square"*, uses the square of the first-stage residuals as the dependent variable. Default: *stage3=NULL*.
stage3_xlist	List of explanatory variables for heteroskedasticity. By default, the single variable *timesale = time1-time0* is constructed and used as the explanatory variable when *stage3="abs"* or *stage3="square"*. Alternatively, a formula can be provided for a user-specified list of explanatory variables, e.g., *stage3_xlist=~x1+x2*. *Important*: note the "~" before the variable list.
print	If *print=T*, prints the regression results. Prints one stage only – the first stage when *stage=NULL* and the final stage when *stage3="square"* or *stage3="abs"*. Default: *print=T*.

Details

The repeat sales model is

$$y_t - y_s = \delta_t - \delta_s + u_t - u_s$$

where *y* is the log of sales price, *s* denotes the earlier sale in a repeat sales pair, and *t* denotes the later sale. Each entry of the data set should represent a repeat sales pair, with $price0 = y_s$, $price1 = y_t$, $time0 = s$, and $time1 = t$. The function *repsaledata* can help transfer a standard hedonic data set to a set of repeat sales pairs.

Repeat sales estimates are sometimes very sensitive to sales from the first few time periods, particularly when the sample size is small. The option *mergefirst* indicates the number of time periods for which the price index is constrained to equal zero. The default is *mergefirst = 1*, meaning that the price index equals zero for just the first time period. The *repsale* command does not have an option for including an intercept in the model.

Following Case and Shiller (1987), many authors use a three-stage procedure to construct repeat sales price indexes that are adjusted for heteroskedasticity related to the length of time between sales. Common specifications for the second-stage function are $e^2 = \alpha_0 + \alpha_1(t-s)$ or $|e| = \alpha 0 + \alpha 1(t-s)$, where *e* represents the first-stage residuals. The first equation implies an error variance of $\hat{\sigma^2} = \hat{e^2}$ and the second equation leads to $\hat{\sigma^2} = \left|\hat{e}\right|^2$. The *repsale* function uses a standard *F* test to determine whether the slope cofficients are significant in the second-stage regression. The results are reported if *print=T*.

The third-stage equation is

$$\frac{y_t - y_s}{\hat{\sigma}} = \frac{\delta_t - \delta_s}{\hat{\sigma}} + \frac{u_t - u_s}{\hat{\sigma}}$$

This equation is estimated by regressing $y_t - y_s$ on the series of indicator variables implied by $\delta_t - \delta_s$ using the *weights* option in *lm* with *weights* = $1/\hat{\sigma^2}$

Value

`fit`	Full regression model.
`pindex`	The estimated price index.
`lo`	The lower bounds for the price index confidence intervals.
`hi`	The upper bounds for the price index confidence intervals.
`dy`	The dependent variable for the repeat sales regression, *dy = price1-price0.*
`xmat`	The matrix of explanatory variables for the repeat sales regressions. *dim(xmat)* = *nt - mergefirst,* where *nt* = the number of time periods and *mergefirst* is specified in the call to *repsale.*

References

Case, Karl and Robert Shiller, "Prices of Single-Family Homes since 1970: New Indexes for Four Cities," *New England Economic Review* (1987), 45-56.

See Also

repsaledata

repsalefourier

repsaleqreg

Examples

```
set.seed(189)
n = 2000
# sale dates range from 0-10
# drawn uniformly from all possible time0, time1 combinations with time0<time1
tmat <- expand.grid(seq(0,10), seq(0,10))
tmat <- tmat[tmat[,1]<tmat[,2], ]
tobs <- sample(seq(1:nrow(tmat)),n,replace=TRUE)
time0 <- tmat[tobs,1]
time1 <- tmat[tobs,2]
timesale <- time1-time0
table(timesale)

# constant variance; index ranges from 0 at time 0 to 1 at time 10
y0 <- time0/10 + rnorm(n,0,.2)
y1 <- time1/10 + rnorm(n,0,.2)
fit <- repsale(price0=y0, price1=y1, time0=time0, time1=time1)
```

```
# variance rises with timesale
# var(u0) = .2^2; var(u1) = (.2 + timesale/10)^2
# var(u1-u0) = var(u0) + var(u1) = 2*(.2^2) + .4*timesale/10 + (timesale^2)/100
y0 <- time0/10 + rnorm(n,0,.2)
y1 <- time1/10 + rnorm(n,0,.2+timesale/10)
par(ask=TRUE)
fit <- repsale(price0=y0, price1=y1, time0=time0, time1=time1)
summary(fit$pindex)
fit <- repsale(price0=y0, price1=y1, time0=time0, time1=time1, stage3="abs")
summary(fit$pindex)
timesale2 <- timesale^2
fit <- repsale(price0=y0, price1=y1, time0=time0, time1=time1, stage3="square",
  stage3_xlist=~timesale+timesale2)
```

`repsaledata` *Preparation of a Repeat Sales Data Set*

Description

Identifies repeat sales from a data set with observations on sale price, time of sale, and a property id. Returns a data frame in which each observation is a repeat sales pair.

Usage

```
repsaledata(price,timevar,id)
```

Arguments

`price`	Variable representing sale price
`timevar`	Variable representing date of sale
`id`	Property id

Details

The input to *repsaledata* is meant to be a set of variables from a standard hedonic data set – the sale price, date, and an *id* number for the individual property. The function identifies the subset of properties that sold at least twice and forms a new data set in which each observation is a repeat sales pair, with "0" denoting the earlier time and "1" denoting the later date in the variable names. The observations are ordered first by id, then by timevar, and then by price. A repeat sales pair is formed by matching an observation for which id(t)=id(t-1) and timevar(t)=timevar(t-1). Thus, a property that sold in times 1, 2, and 3 will produce 2 repeat sales pairs: (1) t = 1 and 2, and (2) t = 2 and 3.

The output of *repsaledata* is a data frame with 5 variables. If some of the original hedonic data set variables need to be included in the repeat sales data set, the original hedonic data set and the *repsaledata* data frame can be merged by the *id* variable.

Value

id	Property id
price0	Sale price at earlier date
time0	Earlier sales date
price1	Sale price at later date
time1	Later sales date

See Also

repsale

repsalefourier

repsaleqreg

Examples

```
id <-      c(1,1,1, 2,2,2, 3,3,3, 4,4,4, 5,5,5)
timevar <- c(1,2,3, 1,2,2, 3,1,1, 1,1,2, 2,2,3)
price <- seq(1:15)
basedata <- data.frame(id,timevar,price)
basedata

rdata <- repsaledata(price,timevar,id)
rdata
```

---

repsalefourier *Repeat Sales Estimation using Fourier Expansions*

Description

Standard and Weighted Least Squares Repeat Sales Estimation using Fourier Expansions

Usage

```
repsalefourier(price0,time0,price1,time1,mergefirst=1,q=1, graph=TRUE,
  graph.conf=TRUE,conf=.95,stage3=FALSE,stage3_xlist=~timesale,
  print=TRUE)
```

Arguments

price0	Earlier price in repeat sales pair
time0	Earlier time in repeat sales pair
price1	Later price in repeat sales pair
time1	Later time in repeat sales pair

mergefirst	Number of initial periods with coefficients constrained to zero. Default: *mergefirst=1*
q	Sets *Q* for the fourier expansion. Default: *q*=1.
graph	If TRUE, graph results. Default: *graph=T*
graph.conf	If TRUE, add confidence intervals to graph. Default: *graph.conf=T*
conf	Confidence level for intervals. Default: .95
stage3	If *stage3 = NULL*, no corrections for heteroskedasticty. If *stage3="abs"*, uses the absolute value of the first-stage residuals as the dependent variable in the second-stage regression. If *stage3="square"*, uses the square of the first-stage residuals as the dependent variable. Default: *stage3=NULL*.
stage3_xlist	List of explanatory variables for heteroskedasticity. By default, the single variable *timesale = time1-time0* is constructed and used as the explanatory variable when *stage3="abs"* or *stage3="square"*. Alternatively, a formula can be provided for a user-specified list of explanatory variables, e.g., *stage3_xlist=~x1+x2*. *Important*: note the "~" before the variable list.
print	If *print=T*, prints the regression results. Prints one stage only – the first stage when *stage=NULL* and the final stage when *stage3="square"* or *stage3="abs"*. Default: *print=T*.

Details

The repeat sales model is

$$y_t - y_s = \delta_t - \delta_s + u_t - u_s$$

where *y* is the log of sale price, *s* denotes the earlier sale in a repeat sales pair, and *t* denotes the later sale. Each entry of the data set should represent a repeat sales pair, with $price0 = y_s$, $price1 = y_t$, $time0 = s$, and $time1 = t$. The function *repsaledata* can help transfer a standard hedonic data set to a set of repeat sales pairs.

The repeat sales model can be derived from a hedonic price function with the form $y_{i,t} = \delta_t + X_i\beta + u_{i,t}$ where X_i is a vector of variables that are assumed constant over time. *repsalefourier* replaces δ_t with a smooth continuous function, $g(T_i)$ where T_i denotes the time of sale for observation *i*. Letting $g(T_i) = \alpha_0 + \alpha_1 z_i + \alpha_2 z_i^2 + \sum_{i=1}^{Q}\{\lambda_q sin(qz_i) + \gamma_q cos(qz_i)\}$, where $z_i = 2\pi(T_i - min(T_i))/(max(T_i) - min(T_i))$, the repeat sales model becomes $y_{i,t} - y_{i,s} = g(T_i) - g(T_i^s) =$

$$\alpha_1(z_i - z_i^s) + \alpha_2(z_i^2 - z_i^{s2}) + \sum_{q=1}^{Q}\{\lambda_q(sin(qz_i) - sin(qz_i^s)) + \gamma_q(cos(qz_i) - cos(z_i^s))\} + u_{i,t} - u_{i,t-s}$$

After imposing the constraint that the price index in the base time period equals zero, the index is constructed from the estimated regression using the following expression:

$$g(T_i) = \alpha_1 z_i + \alpha_2 z_i^2 + \sum_{q=1}^{Q}\{\lambda_q sin(qz_i) + \gamma_q(cos(qz_i) - 1)\}$$

More details can be found in McMillen and Dombrow (2001).

Repeat sales estimates are sometimes very sensitive to sales from the first few time periods, particularly when the sample size is small. The option *mergefirst* indicates the number of time periods for

which the price index is constrained to equal zero. The default is *mergefirst = 1*, meaning that the price index equals zero for just the first time period. The *repsalefourier* command does not have an option for including an intercept in the model.

Following Case and Shiller (1987), many authors use a three-stage procedure to construct repeat sales price indexes that are adjusted for heteroskedasticity related to the length of time between sales. Common specifications for the second-stage function are $e^2 = \alpha_0 + \alpha_1(t-s)$ or $|e| = \alpha 0 + \alpha 1(t-s)$, where *e* represents the first-stage residuals. The first equation implies an error variance of $\hat{\sigma^2} = \hat{e^2}$ and the second equation leads to $\hat{\sigma^2} = \hat{|e|}^2$. The *repsalefourier* function uses a standard *F* test to determine whether the slope cofficients are significant in the second-stage regression. The results are reported if *print=T*.

The third-stage equation is

$$\frac{y_t - y_s}{\hat{\sigma}} = \frac{g(T_i) - g(T_i^s)}{\hat{\sigma}} + \frac{u_t - u_s}{\hat{\sigma}}$$

This equation is estimated by regressing $y_t - y_s$ on $z, z^2, sin(z)...sin(Qz), cos(z)...cos(Qz)$ using the *weights* option in *lm* with *weights* = $1/\hat{\sigma^2}$

Value

`fit`	Full regression model.
`pindex`	The estimated price index.
`lo`	The lower bounds for the price index confidence intervals.
`hi`	The upper bounds for the price index confidence intervals.
`dy`	The dependent variable for the repeat sales regression, *dy = price1-price0.*
`xmat`	The matrix of explanatory variables for the repeat sales regressions. $dim(xmat) = 2 + 2Q$.

References

Case, Karl and Robert Shiller, "Prices of Single-Family Homes since 1970: New Indexes for Four Cities," *New England Economic Review* (1987), 45-56.

McMillen, Daniel P. and Jonathan Dombrow, "A Flexible Fourier Approach to Repeat Sales Price Indexes," *Real Estate Economics* 29 (2001), 207-225.

See Also

repsale

repsaledata

repsaleqreg

Examples

```
set.seed(189)
n = 2000
# sale dates range from 0-50
# drawn uniformly from all possible time0, time1 combinations with time0<time1
tmat <- expand.grid(seq(0,50), seq(0,50))
tmat <- tmat[tmat[,1]<tmat[,2], ]
tobs <- sample(seq(1:nrow(tmat)),n,replace=TRUE)
time0 <- tmat[tobs,1]
time1 <- tmat[tobs,2]
timesale <- time1-time0
timesale2 <- timesale^2

par(ask=TRUE)
z0 <- 2*pi*time0/50
z0sq <- z0^2
sin0 <- sin(z0)
cos0 <- cos(z0)
z1 <- 2*pi*time1/50
z1sq <- z1^2
sin1 <- sin(z1)
cos1 <- cos(z1)
ybase0 <- z0 + .05*z0sq -.5*sin0 - .5*cos0
miny <- min(ybase0)
ybase0 <- ybase0-miny
ybase1 <- z1 + .05*z1sq -.5*sin1 - .5*cos1 - miny
maxy <- max(ybase1)
ybase0 <- ybase0/maxy
ybase1 <- ybase1/maxy
summary(data.frame(ybase0,ybase1))
sig1 = sd(c(ybase0,ybase1))/2
y0 <- ybase0 + rnorm(n,0,sig1)
y1 <- ybase1 + rnorm(n,0,sig1)
fit <- lm(y0~z0+z0sq+sin0+cos0)
summary(fit)
plot(time0,fitted(fit))
fit <- lm(y1~z1+z1sq+sin1+cos1)
summary(fit)
plot(time1,fitted(fit))

fit1 <- repsale(price1=y1,price0=y0,time1=time1,time0=time0,graph=FALSE,
  mergefirst=5)
fit2 <- repsalefourier(price1=y1,price0=y0,time1=time1,time0=time0,q=1,
  graph=FALSE,mergefirst=5)
timevar <- seq(0,50)
plot(timevar,fit1$pindex,type="l",xlab="Time",ylab="Index",
  ylim=c(min(fit1$pindex),max(fit2$pindex)))
lines(timevar,fit2$pindex)



# variance rises with timesale
```

```
# var(u0) = sig1^2; var(u1) = (sig1 + timesale/50)^2
# var(u1-u0) = var(u0) + var(u1) = 2*(sig1^2) + 2*sig1*timesale/10 + (timesale^2)/2500
y0 <- ybase0 + rnorm(n,0,sig1)
y1 <- ybase1 + rnorm(n,0,sig1+timesale/50)
par(ask=TRUE)
fit1 <- repsalefourier(price0=y0, price1=y1, time0=time0, time1=time1,
  graph=FALSE)
fit2 <- repsalefourier(price0=y0, price1=y1, time0=time0, time1=time1,
  graph=FALSE,stage3="abs",stage3_xlist=~timesale+timesale2)
plot(timevar,fit1$lo,type="l",xlab="Time",ylab="Index",
  ylim=c(min(fit1$lo,fit2$lo),max(fit1$hi,fit2$hi)))
lines(timevar,fit1$hi)
lines(timevar,fit2$lo,col="red")
lines(timevar,fit2$hi,col="red")
```

`repsaleqreg` *Quantile Repeat Sales Estimation*

Description

Median-Based Repeat Sales Estimation

Usage

```
repsaleqreg(price0,time0,price1,time1,mergefirst=1,
  graph=TRUE,graph.conf=TRUE,conf=.95,print=TRUE)
```

Arguments

`price0`	Earlier price in repeat sales pair
`time0`	Earlier time in repeat sales pair
`price1`	Later price in repeat sales pair
`time1`	Later time in repeat sales pair
`mergefirst`	Number of initial periods with coefficients constrained to zero. Default: *mergefirst=1*
`graph`	If TRUE, graph results. Default: *graph=T*
`graph.conf`	If TRUE, add confidence intervals to graph. Default: *graph.conf=T*
`conf`	Confidence level for intervals. Default: .95
`print`	If *print=T*, prints the regression results. Default: *print=T*.

Details

The repeat sales model is

$$y_t - y_s = \delta_t - \delta_s + u_t - u_s$$

where *y* is the log of sales price, *s* denotes the earlier sale in a repeat sales pair, and *t* denotes the later sale. Each entry of the data set should represent a repeat sales pair, with $price0 = y_s$, $price1 = y_t$, $time0 = s$, and $time1 = t$. The function *repsaledata* can help transfer a standard hedonic data set to a set of repeat sales pairs.

Repeat sales estimates are sometimes very sensitive to sales from the first few time periods, particularly when the sample size is small. The option *mergefirst* indicates the number of time periods for which the price index is constrained to equal zero. The default is *mergefirst = 1*, meaning that the price index equals zero for just the first time period.

The *repsaleqreg* function uses the *quantreg* package to estimate a quantile regression for the .50 quantile, i.e., the median. A median-based estimator is less sensitive to outliers than linear regression. McMillen and Thorsnes (2006) show that the quantile approach is less sensitive to the inclusion of properties that have undergone renovations between sales. *repsaleqreg* first fits a standard quantile model, including the interecpt. The coefficient vector is then rotated to have a zero intercept using the formula for transforming unrestricted linear regression estimates to the restricted (zero intercept) values:

```
fit <- rq(dy~x)
b <- fit$coef
fit1 <- summary(fit,covariance=TRUE)
vmat <- fit1$cov
k = length(b1)
rmat <- diag(k)
rmat[,1] <- rmat[,1] - vmat[1,]/vmat[1,1]
bmat <- rmat
```

Value

`fit`	Full quantile regression model.
`pindex`	The estimated price index.
`lo`	The lower bounds for the price index confidence intervals.
`hi`	The upper bounds for the price index confidence intervals.

References

Case, Karl and Robert Shiller, "Prices of Single-Family Homes since 1970: New Indexes for Four Cities," *New England Economic Review* (1987), 45-56.

McMillen, Daniel P. and Paul Thorsnes, "Housing Renovations and the Quantile Repeat Sales Price Index," *Real Estate Economics* 34 (2006), 567-587.

See Also

repsale

repsaledata

repsalefourier

Examples

```
set.seed(189)
n = 2000
# sale dates range from 0-10
# drawn uniformly from all possible time0, time1 combinations with time0<time1
tmat <- expand.grid(seq(0,10), seq(0,10))
tmat <- tmat[tmat[,1]<tmat[,2], ]
tobs <- sample(seq(1:nrow(tmat)),n,replace=TRUE)
time0 <- tmat[tobs,1]
time1 <- tmat[tobs,2]
timesale <- time1-time0
table(timesale)

# constant variance; index ranges from 0 at time 0 to 1 at time 10
y0 <- time0/10 + rnorm(n,0,.2)
y1 <- time1/10 + rnorm(n,0,.2)
fit <- repsaleqreg(price0=y0, price1=y1, time0=time0, time1=time1)

# variance rises with timesale
# var(u0) = .2^2; var(u1) = (.2 + timesale/10)^2
# var(u1-u0) = var(u0) + var(u1) = 2*(.2^2) + .4*timesale/10 + (timesale^2)/100
y0 <- time0/10 + rnorm(n,0,.2)
y1 <- time1/10 + rnorm(n,0,.2+timesale/10)
par(ask=TRUE)
fit <- repsaleqreg(price0=y0, price1=y1, time0=time0, time1=time1)
summary(fit$pindex)
```

---

sarml *Spatial AR Maximum-Likelihood Estimation*

Description

Estimates the model $Y = \rho WY + X\beta + u$ by maximizing the log-likelihood function.

Usage

```
sarml(form,wmat=NULL,shpfile=NULL,wy=NULL,eigvar=NULL,startrho=NULL,
  print=TRUE,data=NULL)
```

Arguments

form	Model formula
wmat	The *W* matrix. If not specified, *W* will be calculated from the shape file. Default: *W* = NULL.
shpfile	Shape file. Needed unless (1) *wy* and *eigvar* are both provided, or (2) *wmat* and *eigvar* are provided

wy	The *WY* variable. Default: not specified; program attempts to calculate *WY* using *wmat* or *shpfile*.
eigvar	The vector of eigenvalues for *W*. Default: not provided. *shpfile* must be specified to calculate the eigenvalues within the *sarml* command.
startrho	A starting value for ρ. Default: *startrho=0.* Estimation will generally be faster if *startrho = 0.*
print	If *print=F*, no results are printed. Default: *print=T.*
data	A data frame containing the data. Default: use data in the current working directory

Details

The primary motivation for the *sarml* command is to provide a convenient way to estimate multiple spatial AR models without having to calculate the eigenvalues of *W* each time. Under the assumption that the errors, *u*, are independently and identically distributed normal, the log-likelihood function for the spatial AR model is

$$lnl = -\frac{n}{2}log(\pi) - \frac{n}{2}log(\sigma^2) - \frac{1}{2\sigma^2}\sum_i u_i^2 + \sum_i log(1 - \rho * eigvar_i)$$

where *eigvar* is the vector of eigenvalues of *W*. Though *spdep* provides a convenient and fast method for calculating the eigenvalues from a shape file, the calculation can nonetheless take a very long time for large data sets. The *sarml* command allows the user to input the vector of eigenvalues directly, which saves time when several models are estimated using the same *W* matrix. Unless a vector of eigenvalues is provided using the *eigvar* option, the eigenvalues are calculated from the shape file (provided using the *shpfile* option) using the *spdep* package.

Conditional on the value of ρ, the maximum likelihood estimate of β is simply the vector of coefficients from a regression of $Y - \rho WY$ on X. The estimate of the error variance also has a closed form solution: $\hat{\sigma^2} = \sum_i u_i^2/n$. Substituting these estimates into the log-likelihood functions leads to the following concentrated log-likelihood function:

$$lc = -\frac{n}{2}(log(\pi) + 1) - \frac{n}{2}log(\sum_i u_i^2) + \sum_i log(1 - \rho * eigvar_i)$$

Working with the concentrated likelihood function reduces the optimization problem to a one-dimensional search for the value of ρ that maximizes *lc*. Unless a value is provided for *startrho*, the *sarml* procedure begins by using the *optimize* command to find the value of ρ that maximizes *lc*. This estimate of ρ (or the value provided by the *startrho* option) is then used to calculate the implied values of β and σ^2, and these values are used as starting values to maximize *lnl* using the *nlm* command.

The covariance matrix for the estimates of β and ρ, *vmat*, is the inverse of $(1/\sigma^2)V$. *V* has partitions $V_{11} = X'X$, $V_{12} = X'WY$, $V_{21} = Y'W'X$, and $V_{22} = Y'W'WY + \sigma^2 \sum_i \frac{eigvar_i}{(1-\rho * eigvar_i)^2}$.

Value

beta	The estimated vector of coefficients, β.

rho	The estimated value of ρ.
sig2	The estimated error variance, σ^2.
vmat	The covariance matrix for (β, ρ^2).
eigvar	The vector of eigenvalues

See Also

makew

qregspiv

qregbmat

qregsim1

qregsim2

qregcpar

qreglwr

Examples

```
library(spdep)

cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1&cmap$CAREA!="O'Hare",]
samppop <- cmap$POPULATION>0&cmap$AREA>0
cmap <- cmap[samppop,]
cmap$lndens <- log(cmap$POPULATION/cmap$AREA)
lmat <- coordinates(cmap)
cmap$LONGITUDE <- lmat[,1]
cmap$LATITUDE  <- lmat[,2]
cmap$dcbd <- geodistance(longvar=cmap$LONGITUDE,latvar=cmap$LATITUDE,
  lotarget=-87.627800,latarget=41.881998)$dist

fit <- makew(shpfile=cmap,eigenvalues=TRUE)
wmat <- fit$wmat
eigvar <- fit$eigvar

# input w, calculate eigvar within sarml
fit <- sarml(lndens~dcbd,wmat=wmat,eigvar=eigvar,data=cmap)
```

---

semip *Semi-Parametric Regression*

Description

Estimates a semi-parametric model with the form $y = X\beta + f(z) + u$, where f(z) is either fully nonparametric with $f(z) = f(z_1)$ or conditionally parametric with $f(z) = z_2\lambda(z_1)$.

Usage

```
semip(form,nonpar,conpar,window1=.25,window2=.25,bandwidth1=0,bandwidth2=0,
 kern="tcub",distance="Mahal",targetfull=NULL, print.summary=TRUE, data=NULL)
```

Arguments

form	Model formula. Specifies the base parametric form of the model, $y = X\beta$. Any number of variables can be included in X. Format: semip(y~x1+x2..., ...).
nonpar	List of variables in z_1. Formats: semip(..., nonpar=~z1a, ...) or semip(..., nonpar=~z1a+zb, ...). Important: note the "~" before the first z1 variable. At most two variables can be included in z_1.
conpar	List of variables in z_2. By default, *conpar* = NULL and f(z) has the fully nonparametric form $f(z) = f(z_1)$; in this case the variables in z_1 are taken from the list provided by *nonpar*. If a list of variables is provided for *nonpar*, the conditionally parametric form $f(z) = z_2\lambda(z_1)$ is assumed for f(z), and the variables for z_2 are provided by *conpar*. Any number of variables can be included in *conpar*. Format: semip(..., conpar=~z2a+z2b+z2c+..., ...). Important: note the "~" before the first *z2* variable.
window1	Window size for the LWR or CPAR regressions of *y* and *x* on *z*. Default = .25.
window2	Window size for the LWR or CPAR regression of $y - X\hat{\beta}$ on *z*. Default = .25.
bandwidth1	Bandwidth for the LWR or CPAR regressions of *y* and *x* on *z*. Default: not specified.
bandwidth2	Bandwidth for the LWR or CPAR regression of $y - X\hat{\beta}$ on *z*. Default: not specified.
kern	Kernel weighting functions. Default is the tri-cube. Options include "rect", "tria", "epan", "bisq", "tcub", "trwt", and "gauss".
distance	Options: "Euclid", "Mahal", or "Latlong" for Euclidean, Mahalanobis, or "great-circle" geographic distance. May be abbreviated to the first letter but must be capitalized. Note: *semip* looks for the first two letters to determine which variable is latitude and which is longitude, so data set must be attached first or specified using the data option; options like data$latitude will not work. Default: Mahal.
targetfull	Target options to be passed to the *lwr* command if *conpar* = *NULL* or the *cparlwr* command if a list of variables is provided for *conpar*. Options include *NULL*, "alldata", or the *full* output of the *maketarget* command. The appropriate argument will then be passed on to the *lwr* or *cparlwr* command.
print.summary	If *print.summary=T*, prints a summary of the regression results for *ey* on *ex*, i.e., the parametric portion of the model. Default: *print.summary=T*.
data	A data frame containing the data. Default: use data in the current working directory

Details

If *conpar = NULL*, the function implements Robinson's (1988) semi-parametric estimator for the model $y = X\beta + f(z) + u$. In this case, the list of variables in z is taken from *nonpar* and z can have at most two variables. If a list of variables is provided for *conpar*, the function implements the semi-parametric estimator for the model $f(z) = z_2\lambda(z_1)$. In this case, the list of variables in $z1$ is taken from *nonpar* and the list of variables in z_2 is taken from *conpar*. z_1 can have at most two variables. There is no limit on the number of variables in z_2.

The estimation procedure has the following three steps under either specification:

1. Nonparametric regressions of *y* on *z* and each *X* on *z* using the *lwr* function when *conpar=NULL* and the *cparlwr* function when a list of variables is provided for *cparlwr*. The window or bandwidth for these regressions is set by *window1* or *bandwidth1*.

2. OLS regression of $y - \hat{y}$ on the *k-1* variables in $X - \hat{X}$, omitting the intercept. The coefficients from this regression are the estimated values of β.

3. Nonparametric regression of $y - X\hat{\beta}$ on *z* using the *lwr* function when *conpar=NULL* and the *cparlwr* function when a list of variables is provided for *cparlwr*. The window or bandwidth for these regressions is set by *window2* or *bandwidth2*.

The stage-two OLS regressions use *k* degrees of freedom. The stage-three nonparametric regression uses *2*df1-df2* degrees of freedom, where $df1 = tr(L)$ and $df2 = tr(L'L)$ and *L* is the *nxn* matrix for the *lwr* or *cparlwr* regression $L(Y - X\hat{\beta})$. The estimated variance is $\hat{\sigma^2} = rss/(n - 2 * df1 + df2)$, where $rss = \sum_i (y_i - X_i\beta - f(z_i))^2$. The covariance matrix estimate for β is $\hat{\sigma^2}((X - \hat{X})'(X - \hat{X}))^{-1}$. The covariance matrix is stored as *vmat*.

The nonparametric regressions are estimated using either the *lwr* or *cparlwr* function. See their descriptions for more information.

Value

`xcoef`	The estimated coefficients for the parametric part of the model, β.
`vmat`	The covariance matrix for the estimates of β.
`xbhat`	The predicted values of *y* for the full data set.
`nphat`	The predicted values of *f(z)* for the full data set. *mean(xbhat)+mean(nphat)* will be close but not necessarily identical to *mean(y)*.
`nphat.se`	Standard errors for the predicted values of *y* for the full data set.
`npfit`	The complete set of *lwr* or *cparlwr* results from the nonparametric regression of $y - X\beta$ on Z.
`df1`	*k* + *tr(L)*, where *k* is the number of explanatory variables in $X\beta$ (including the constant) and *L* is the *nxn* matrix used to calculate the final-stage nonparametric or conditionally parametric regression of $Y - X\hat{\beta}$ on *Z*. *df1* is one measure of the degrees of freedom used in estimation.
`df2`	An alternative measure of the degrees of freedom used in estimation, $df2 = k + tr(L'L)$.
`sig2`	Estimated residual variance, $sig2 = rss/(n - 2 * df1 + df2)$.

References

Cleveland, William S. and Susan J. Devlin, "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting," *Journal of the American Statistical Association* 83 (1988), 596-610.

Loader, Clive. *Local Regression and Likelihood*. New York: Springer, 1999.

McMillen, Daniel P., "Issues in Spatial Data Analysis," *Journal of Regional Science* 50 (2010), 119-141.

McMillen, Daniel P., "Employment Densities, Spatial Autocorrelation, and Subcenters in Large Metropolitan Areas," *Journal of Regional Science* 44 (2004), 225-243.

McMillen, Daniel P. and Christian Redfearn, "Estimation and Hypothesis Testing for Nonparametric Hedonic House Price Functions," *Journal of Regional Science* 50 (2010), 712-733.

Pagan, Adrian and Aman Ullah. *Nonparametric Econometrics*. New York: Cambridge University Press, 1999.

Robinson, Paul M. 1988. "Root-N-Consistent Semiparametric Regression," *Econometrica*, 56, 931-954.

See Also

cparlwr

lwr

maketarget

Examples

```
# Single variable in f(z)
par(ask=TRUE)
n = 1000
x <- runif(n,0,2*pi)
x <- sort(x)
z <- runif(n,0,2*pi)
xsq <- x^2
sinx <- sin(x)
cosx <- cos(x)
sin2x <- sin(2*x)
cos2x <- cos(2*x)
ybase1 <-  x - .1*xsq + sinx - cosx - .5*sin2x + .5*cos2x
ybase2 <- -z + .1*(z^2) - sin(z) + cos(z) + .5*sin(2*z) - .5*cos(2*z)
ybase <- ybase1+ybase2
sig = sd(ybase)/2
y <- ybase + rnorm(n,0,sig)

# Correct specification for x; z in f(z)
fit <- semip(y~x+xsq+sinx+cosx+sin2x+cos2x,nonpar=~z,window1=.20,window2=.20)
2*fit$df1 - fit$df2
yvect <- c(min(ybase1,fit$xbhat), max(ybase1, fit$xbhat))
xbhat  <- fit$xbhat - mean(fit$xbhat) + mean(ybase1)
```

```
plot(x,ybase1,type="l",xlab="x",ylab="ybase1",ylim=yvect, main="Predictions for XB")
lines(x, xbhat, col="red")

predse <- sqrt(fit$sig2 + fit$nphat.se^2)
nphat <- fit$nphat - mean(fit$nphat) + mean(ybase2)
lower <- nphat + qnorm(.025)*fit$nphat.se
upper <- nphat + qnorm(.975)*fit$nphat.se
o <- order(z)
yvect <- c(min(lower), max(upper))
plot(z[o], ybase2[o], type="l", xlab="z", ylab="f(z) ",
   main="Predictions for f(z) ", ylim=yvect)
lines(z[o], nphat[o], col="red")
lines(z[o], lower[o], col="red", lty="dashed")
lines(z[o], upper[o], col="red", lty="dashed")

## Not run:
# Chicago Housing Sales
data(matchdata)
match05 <- data.frame(matchdata[matchdata$year==2005,])
match05$age <- 2005-match05$yrbuilt

tfit1 <- maketarget(~dcbd,window=.3,data=match05)
tfit2 <- maketarget(~longitude+latitude,window=.5,data=match05)

# nonparametric control for dcbd

fit <- semip(lnprice~lnland+lnbldg+rooms+bedrooms+bathrooms+centair+fireplace+brick+
garage1+garage2+ age+rr, nonpar=~dcbd, data=match05,targetfull=tfit1)

# nonparametric controls for longitude and latitude

fit <- semip(lnprice~lnland+lnbldg+rooms+bedrooms+bathrooms+centair+fireplace+brick+
garage1+garage2+ age+rr+dcbd, nonpar=~longitude+latitude, data=match05, targetfull=tfit2,
distance="Latlong")

# Conditionally parametric model:  y = XB + dcbd*lambda(longitude,latitude) + u
fit <- semip(lnprice~lnland+lnbldg+rooms+bedrooms+bathrooms+centair+fireplace+
 brick+garage1+garage2+age+rr, nonpar=~longitude+latitude, conpar=~dcbd,
 data=match05, distance="Latlong",targetfull=tfit1)

# Conditional parametric model:  y = XB + Z*lambda(longitude,latitude) + u
# Z = (dcbd,lnland,lnbldg,age)
fit <- semip(lnprice~rooms+bedrooms+bathrooms+centair+fireplace+brick+
garage1+garage2+rr, nonpar=~longitude+latitude, conpar=~dcbd+lnland+lnbldg+age,
 data=match05, distance="Latlong",targetfull=tfit2)

## End(Not run)
```

---

smooth12 *Univariate or Bivariate Interpolation*

Description

Uses the Akima (1970) method for univariate interpolation and the Modified Shephard Algorithm for bivariate interpolation.

Usage

```
smooth12(x,y,xout,knum=16,std=TRUE)
```

Arguments

x	The actual values of the x-variable(s). A simple numeric variable for univariate interpolation and a matrix of locations for bivariate interpolation.
y	The variable to be interpolated.
xout	Points on the x-axis where the function is to be evaluated. A single numeric variable in the case of univariate interpolation and a matrix of locations for bivariate interpolation.
knum	The number of target points used for bivariate interpolation.
std	If *TRUE*, re-scales the columns of *x* and *xout* by dividing by the standard deviation of the columns in *x*. Not applicable for univariate interpolation.

Details

The univariate version of the function is designed as a partial replacement for the *aspline* function in the *akima* package. It produces a smooth function that closely resembles the interpolation that would be done by hand. Values of y are averaged across any ties for x. The function does not allow for extrapolation beyond the min(x), max(x) range.

The bivariate version of the function uses the modifed Shepard's method for interpolation. The function uses the RANN package to find the nearest *knum* target points to each location in *xout*. The following formula is used to interpolate from these target points to the locations given in *xout*:

$$\frac{\sum_{i=1}^{knum} w_i y_i}{\sum_{i=1}^{knum} w_i}$$

where

$w_i = ((maxd - d_i)^2)/(maxd * d_i)$

and

$maxd = max(d_1, ..., d_{knum})$.

Value

The values of y interpolated to the *xout* locations.

References

Akima, Hiroshi, "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures," *Journal of the Association for Computing Machinery* 17 (1970), 589-602.

Franke, R. and G. Neilson, "Smooth Interpolation of Large Sets of Scatter data," *International Journal of Numerical Methods in Engineering* 15 (1980), 1691 - 1704.

See Also

maketarget

Examples

```
set.seed(484849)
n = 1000
x <- runif(n,-2*pi,2*pi)
x <- sort(x)
y <- sin(x) + cos(x) - .5*sin(2*x) - .5*cos(2*x) + sin(3*x)/3 + cos(3*x)/3
x1 <- seq(-2*pi,2*pi,length=100)
y1 <- sin(x1) + cos(x1) - .5*sin(2*x1) - .5*cos(2*x1) + sin(3*x1)/3 + cos(3*x1)/3

yout <- smooth12(x1,y1,x)
plot(x,y,type="l")
lines(x,yout,col="red")

x <- seq(0,10)
xmat <- expand.grid(x,x)
y <- sqrt((xmat[,1]-5)^2 + (xmat[,2]-5)^2)
xout <- cbind(runif(n,0,10),runif(n,0,10))
y1 <- sqrt((xout[,1]-5)^2 + (xout[,2]-5)^2)
y2 <- smooth12(xmat,y,xout)
cor(y1,y2)
```

---

splogit *Linearized GMM spatial logit*

Description

Implements the Klier-McMillen (2008) linearized GMM logit model for a 0-1 dependent variable and an underlying latent variable of the form $Y^* = \rho W Y^* + X\beta + u$

Usage

```
splogit(form,inst=NULL,winst=NULL,wmat=NULL,shpfile=NULL,blockid=NULL,
        minblock=NULL,maxblock=NULL,data=NULL,silent=FALSE,minp=NULL)
```

Arguments

form	Model formula
inst	List of instruments *not* to be pre-multiplied by *W*. Entered as *inst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
winst	List of instruments to be pre-multiplied by *W* before use. Entered as *winst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.

wmat	Directly enter *wmat* rather than creating it from a shape file. Default: not specified. One of the *wmat* or *shpfile* options must be specified.
shpfile	Shape file to be used for creating the *W* matrix. Default: not specified. One of the *wmat* or *shpfile* options must be specified. The order of the observations in *wmat* must be the same as the order of observations in *data*.
blockid	A variable identifying groups used to specify a block diagonal structure for the *W* matrix, e.g., *blockid=state* or *blockid=region*. Calculates a separate *W* matrix for each block. The *shpfile* option must be specified; *wmat* is ignored.
minblock	Groups with fewer than *minblock* observations are omitted. Default is the number of explanatory variables, including *WXB*. This option helps to avoid singularity since the instrumental variables are constructed by a separate regression for each block.
maxblock	Groups with more than *maxblock* observations are omitted. Unlimited by default. This option may be useful for very large data sets as full nblock x nblock matrices must be constructed for each block, where nblock is the number of observations in the block.
data	A data frame containing the data. Default: use data in the current working directory
silent	If *silent=T*, no output is printed
minp	Specifies a limit for the estimated probability. Any estimated probability lower than *minp* will be set to *minp* and any probability higher than 1-*minp* will be set to 1-*minp*. By default, the estimated probabilities are bounded by 0 and 1.

Details

The linearized model is a three-step estimation procedure. Let *y* be the indicator value: $y = 1$ when $y^* > 0$ and $y = 0$ when $y^* < 0$. The first stage is standard logit of *y* on *X*. The probability estimates from this regression are $p = exp(X\beta)/(1 + exp(X\beta))$. The second/third stage of the procedure is standard 2SLS estimation of $u = y - p + gX\hat{\beta}$ on gX and $gWX\hat{\beta}$ using *Z* as instruments. *g* is the gradient vector, $dp/d\beta$. The covariance matrix (equation 3 in Klier-McMillen, 2008) is estimated using the *car* package. The final estimates minimize $(y-p)'Z(Z'Z)^{-1}Z'(y-p)$ with *p* linearized around $\hat{\beta}$ and $p = 0$.

splogit provides flexibility in specifying the list of instruments. By default, the instrument list includes *X* and *WX*, where *X* is the original explanatory variable list and *W* is the spatial weight matrix. Either *wmat* or *shpfile* must be specified if *inst* and *winst* are set to their default values.

It is also possible to directly specify the full instrument list or to include only a subset of the *X* variables in the list that is to be pre-multiplied by *W*. Let *list1* and *list2* be user-provided lists of the form *list=~z1+z2*. The combinations of defaults (*NULL*) and lists for *inst* produce the following results for *Z*:

1. *inst* = *NULL*, *winst* = *NULL*, and either *shpfile* or *wmat* specified: *Z* = *(X, WX)*
2. *inst* = *list1*, *winst* = *NULL*, and either *shpfile* or *wmat* specified: *Z* = *list1*
3. *inst* = *NULL*, *winst* = *list2*, and either *shpfile* or *wmat* specified: *Z* = *(X, W*list2)*
4. *inst* = *list1*, *winst* = *list2*, and either *shpfile* or *wmat* specified: *Z* = *(list1, W*list2)*

5. *inst* = *list1*, *winst* = *list2*, and both *shpfile* and *wmat* NOT specified: *Z* = *(list1, list2)*

Note that when *inst=list1* and *winst=NULL* it is up to the user to specify at least one variable in *list1* that is not also included in *X*.

The difference between cases (4) and (5) is that the *list2* variables are left unaltered in case (5) rather than being pre-multiplied by *W*. The case (5) option makes it possible to avoid manipulations of large matrices from within *splogit*. The idea is that W*list2 should be calculated prior to running *splogit*, with the variables implied by W*list2 being provided directly to *splogit* using the *winst* option.

Value

`coef`	Coefficient estimates.
`se`	Standard error estimates.
`u`	The generalized error term.
`gmat`	The matrix of gradient terms, G.

References

Klier, Thomas and Daniel P. McMillen, "Clustering of Auto Supplier Plants in the United States: Generalized Method of Moments Spatial Logit for Large Samples," *Journal of Business and Economic Statistics* 26 (2008), 460-471.

See Also

cparlogit

cparprobit

cparmlogit

gmmlogit

gmmprobit

spprobit

spprobitml

Examples

```
set.seed(9947)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1&cmap$CAREA!="O'Hare",]
wmat <- makew(cmap)$wmat
n = nrow(wmat)
rho = .4
x <- runif(n,0,10)
ystar <- as.numeric(solve(diag(n) - rho*wmat)%*%(x + rnorm(n,0,2)))
y <- ystar>quantile(ystar,.4)
fit <- splogit(y~x,  wmat=wmat)
```

spprobit *Linearized GMM spatial probit*

Description

Implements the Klier-McMillen (2008) linearized GMM probit model for a 0-1 dependent variable and an underlying latent variable of the form $Y^* = \rho WY^* + X\beta + u$

Usage

```
spprobit(form,inst=NULL,winst=NULL,wmat=NULL,shpfile=NULL,blockid=NULL,
         minblock=NULL,maxblock=NULL,data=NULL,silent=FALSE,minp=NULL)
```

Arguments

form	Model formula
inst	List of instruments *not* to be pre-multiplied by *W*. Entered as *inst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
winst	List of instruments to be pre-multiplied by *W* before use. Entered as *winst=~w1+w2 ...* Default: *inst=NULL*. See *details* for more information.
wmat	Directly enter *wmat* rather than creating it from a shape file. Default: not specified.
shpfile	Shape file to be used for creating the *W* matrix. The order of the observations in *wmat* must be the same as the order of observations in *data*.
blockid	A variable identifying groups used to specify a block diagonal structure for the *W* matrix, e.g., *blockid=state* or *blockid=region*. Calculates a separate *W* matrix for each block. The *shpfile* option must be specified; *wmat* is ignored.
minblock	Groups with fewer than *minblock* observations are omitted. Default is the number of explanatory variables, including *WXB*. This option helps to avoid singularity since the instrumental variables are constructed by a separate regression for each block.
maxblock	Groups with more than *maxblock* observations are omitted. Unlimited by default. This option may be useful for very large data sets as full nblock x nblock matrices must be constructed for each block, where nblock is the number of observations in the block.
data	A data frame containing the data. Default: use data in the current working directory
silent	If *silent=T*, no output is printed
minp	Specifies a limit for the estimated probability. Any estimated probability lower than *minp* will be set to *minp* and any probability higher than 1-*minp* will be set to 1-*minp*. By default, the estimated probabilities are bounded by 0 and 1.

Details

The linearized model is a three-step estimation procedure. Let *y* be the indicator value: *y* = 1 when $y^* > 0$ and *y* = 0 when $y^* < 0$. The first stage is standard probit of *y* on *X*. The probability estimates from this regression are $p = \Phi(X\hat{\beta})$ and the generalized error is $e = (y - p)\phi(X\hat{\beta})/(p(1 - p))$. The second/third stage of the procedure is standard 2SLS estimation of $u = e + gX\hat{\beta}$ on gX and $gWX\hat{\beta}$ using *Z* as instruments, where *g* is the gradient vector, $-de/d\hat{\beta}$. The covariance matrix (equation 3 in Klier-McMillen, 2008) is estimated using the *car* package. The final estimates minimize $e'Z(Z'Z)^{-1}Z'e$ with *e* linearized around $\hat{\beta}$ and *p* = 0.

spprobit provides flexibility in specifying the list of instruments. By default, the instrument list includes *X* and *WX*, where *X* is the original explanatory variable list and *W* is the spatial weight matrix. Either *wmat* or *shpfile* must be specified if *inst* and *winst* are set to their default values.

It is also possible to directly specify the full instrument list or to include only a subset of the *X* variables in the list that is to be pre-multiplied by *W*. Let *list1* and *list2* be user-provided lists of the form *list=~z1+z2*. The combinations of defaults (*NULL*) and lists for *inst* produce the following results for *Z*:

1. *inst* = *NULL*, *winst* = *NULL*, and either *shpfile* or *wmat* specified: *Z* = *(X, WX)*

2. *inst* = *list1*, *winst* = *NULL*, and either *shpfile* or *wmat* specified: *Z* = *list1*

3. *inst* = *NULL*, *winst* = *list2*, and either *shpfile* or *wmat* specified: *Z* = *(X, W*list2)*

4. *inst* = *list1*, *winst* = *list2*, and either *shpfile* or *wmat* specified: *Z* = *(list1, W*list2)*

5. *inst* = *list1*, *winst* = *list2*, and both *shpfile* and *wmat* NOT specified: *Z* = *(list1, list2)*

Note that when *inst=list1* and *winst=NULL* it is up to the user to specify at least one variable in *list1* that is not also included in *X*.

The difference between cases (4) and (5) is that the *list2* variables are left unaltered in case (5) rather than being pre-multiplied by *W*. The case (5) option makes it possible to avoid manipulations of large matrices from within *spprobit*. The idea is that W*list2 should be calculated prior to running *spprobit*, with the variables implied by W*list2 being provided directly to *spprobit* using the *winst* option.

Value

`coef`	Coefficient estimates.
`se`	Standard error estimates.
`u`	The generalized error term.
`gmat`	The matrix of gradient terms, G.

References

Klier, Thomas and Daniel P. McMillen, "Clustering of Auto Supplier Plants in the United States: Generalized Method of Moments Spatial Logit for Large Samples," *Journal of Business and Economic Statistics* 26 (2008), 460-471.

See Also

cparlogit

cparprobit

cparmlogit

gmmlogit

gmmprobit

splogit

spprobitml

Examples

```
set.seed(9947)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1&cmap$CAREA!="O'Hare",]
wmat <- makew(cmap)$wmat
n = nrow(wmat)
rho = .4
x <- runif(n,0,10)
ystar <- as.numeric(solve(diag(n) - rho*wmat)%*%(x + rnorm(n,0,2)))
y <- ystar>quantile(ystar,.4)
fit <- spprobit(y~x,  wmat=wmat)
```

---

spprobitml *Maximum Likelihood Estimation of a Spatial Probit Model*

Description

Probit estimation for a model with an underlying latent variable of the form $Y^* = \rho W Y^* + X\beta + u$

Usage

```
spprobitml(form,wmat=NULL,shpfile=NULL,blockid=NULL,
minblock=NULL,maxblock=NULL,stdprobit=TRUE,data=NULL)
```

Arguments

form	Model formula
wmat	Directly enter *wmat* rather than creating it from a shape file. Default: not specified. One of the *wmat* or *shpfile* options must be specified.
shpfile	Shape file to be used for creating the *W* matrix. Default: not specified. One of the *wmat* or *shpfile* options must be specified. The order of the observations in *wmat* must be the same as the order of observations in *data*.

`blockid`	A variable identifying groups used to specify a block diagonal structure for the *W* matrix, e.g., *blockid=state* or *blockid=region*. Calculates a separate *W* matrix for each block. The *shpfile* option must be specified; *wmat* is ignored.
`minblock`	Groups with fewer than *minblock* observations are omitted. Default is the number of explanatory variables, including the spatial lag term.
`maxblock`	Groups with more than *maxblock* observations are omitted. Unlimited by default. This option may be useful for very large data sets as full nblock x nblock matrices must be constructed for each block, where nblock is the number of observations in the block.
`stdprobit`	If *TRUE*, also prints standard probit model results. Default: *stdprobit=TRUE*.
`data`	A data frame containing the data. Default: use data in the current working directory

Details

Estimation is based on the reduced form of the spatial AR model, $Y^* = (I-\rho W)^{-1}(X\beta+u)$. The model structure typically implies heteroskedasticity: the variance of the reduced form error term, $(I-\rho W)^{-1}u$, is $\sigma^2 diag\{(I-\rho W)^{-1}(I-\rho W')^{-1}\}$. For probit estimation, σ^2 is normalized to one. Let s_i^2 denote the variance for observation *i*, and define $X^* = (I-\rho W)^{-1}X$. Then the probability that $Y_i^* > 0$ is $\Phi(X_i^*\beta/s_i)$, and the log-likelihood function is $\sum_i\{y_i ln(\Phi_i) + (1-y_i)ln(1-\Phi_i)\}$. The *spprobitml* commands estimates the model by maximizing this log-likelihood function with respect to β and ρ.

Variants of this approach – maximizing the log-likelihood function implied by the reduced form of the model – were proposed by Case (1992) and McMillen (1992). Case's estimation procedure relies on a simple form of the spatial weight matrix in which each observation within a district is affected equally by the other observations in the district. McMillen's (1992) approach is equivalent to the one used here, but he suggested using an EM algorithm to estimate the model. Neither author suggested a covariance matrix: Case (1992) appears to have relied on the standard probit estimate which applies when the model is estimated conditional on ρ, while McMillen (1992) proposed a bootstrap approach.

A consistent covariance matrix can be calculated using the gradient terms:

$$V(\hat{\theta})^{-1} = \left(\sum_i \partial lnL_i/\partial\hat{\theta}\right)\left(\sum_i \partial lnL_i/\partial\hat{\theta}'\right)$$

The gradient term for $\hat{\rho}$ is calculated using numeric derivatives. The covariance matrix, $V(\hat{\theta})$, is not fully efficient because the estimation procedure only indirectly takes into account the autocorrelation structure. An analogous approach is used to calculate standard errors conditional on $\hat{\rho}$. In the conditional case, only the gradient terms for $\hat{\beta}$ are used; they are evaluated using the estimated values of ρ.

Estimation can be very slow because each iteration requires the inversion of an *nxn* matrix. To speed up the estimation process and to reduce memory requirements, it may be desirable to impose a block diagonal structure on *W*. For example, it may be reasonable to impose that each state or region has its own error structure, with no correlation of errors across regions. The *blockid* option specifies a block diagonal structure such as *blockid=region*. If there are *G* groups, estimation requires *G* submatrices to be inverted rather than one *nxn* matrix, which greatly reduces memory requirements and

significantly reduces the time required in estimation. The weight matrix must be calculated from *shpfile* if the *blockid* option is specified; the *wmat* option should be set to *NULL*.

Value

`coef`	Coefficient estimates.
`logl`	The log-likelihood value.
`vmat1`	The covariance matrix for $\hat{\beta}$, conditional on $\hat{\rho}$.
`vmat2`	The unconditional covariance matrix for $\hat{\theta} = (\hat{\beta}, \hat{\rho})$.

References

Case, Anne C., "Neighborhood Influence and Technological Change," *Regional Science and Urban Economics* 22 (1992), 491-508.

McMillen, Daniel P., "Probit With Spatial Autocorrelation," *Journal of Regional Science* 32 (1992), 335-348.

See Also

cparlogit

cparprobit

cparmlogit

gmmlogit

gmmprobit

makew

splogit

spprobit

Examples

```
set.seed(9947)
cmap <- readShapePoly(system.file("maps/CookCensusTracts.shp",
  package="McSpatial"))
cmap <- cmap[cmap$CHICAGO==1&cmap$CAREA!="O'Hare",]
lmat <- coordinates(cmap)
dnorth <- geodistance(lmat[,1],lmat[,2], lotarget=-87.627800,
latarget=41.881998,dcoor=TRUE)$dnorth
cmap <- cmap[dnorth>1,]
wmat <- makew(cmap)$wmat
n = nrow(wmat)
rho = .4
x <- runif(n,0,10)
ystar <- as.numeric(solve(diag(n) - rho*wmat)%*%(x + rnorm(n,0,2)))
y <- ystar>quantile(ystar,.4)
fit <- spprobitml(y~x,  wmat=wmat)
```

Index