Package 'crs'

February 19, 2015

Version 0.15-24

Date 2014-12-18

Imports boot, stats, np, quantreg, rgl

Suggests logspline, npRmpi, quadprog

Title Categorical Regression Splines

Description

Regression splines that handle a mix of continuous and categorical (discrete) data often encountered in applied settings. I would like to gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada (NSERC:www.nserc.ca), the Social Sciences and Humanities Research Council of Canada (SSHRC:www.sshrc.ca), and the Shared Hierarchical Academic Research Computing Network (SHARCNET:www.sharcnet.ca).

License GPL

URL https://github.com/JeffreyRacine/R-Package-crs/

Author Jeffrey S. Racine [aut, cre],
Zhenghua Nie [aut],
Brian D. Ripley [ctb] (stepCV.R)

Maintainer Jeffrey S. Racine <racinej@mcmaster.ca>

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-12-18 18:37:00

R topics documented:

`crs-package` *Nonparametric Regression Splines with Continuous and Categorical Predictors*

Description

This package provides a method for nonparametric regression that combines the (global) approximation power of regression splines for continuous predictors ('`x`') with the (local) power of kernel methods for categorical predictors ('`z`'). The user also has the option of instead using indicator bases for the categorical predictors. When the predictors contain both continuous and categorical (discrete) data types, both approaches offer more efficient estimation than the traditional sample-splitting (i.e. 'frequency') approach where the data is first broken into subsets governed by the categorical `z`.

To cite the **crs** package type: '`citation("crs")`' (without the single quotes).

For a listing of all routines in the **crs** package type: '`library(help="crs")`'.

For a listing of all demos in the **crs** package type: '`demo(package="crs")`'.

For a '`vignette`' that presents an overview of the **crs** package type: '`vignette("crs")`'.

Details

For the continuous predictors the regression spline model employs the B-spline basis matrix using the B-spline routines in the GNU Scientific Library (`http://www.gnu.org/software/gsl/`).

The `tensor.prod.model.matrix` function is used to construct multivariate tensor spline bases when `basis="tensor"` and uses additive B-splines otherwise (i.e. when `basis="additive"`).

For the discrete predictors the product kernel function is of the 'Li-Racine' type (see Li and Racine (2007) for details) which is formed by constructing products of one of the following univariate kernels:

(z is discrete/nominal) $l(z_i, z, \lambda) = 1$ if $z_i = z$, and λ if $z_i \neq z$. Note that λ must lie between 0 and 1.

(z is discrete/ordinal) $l(z_i, z, \lambda) = 1$ if $|z_i - z| = 0$, and $\lambda^{|z_i - z|}$ if $|z_i - z| \geq 1$. Note that λ must lie between 0 and 1.

Alternatively, for the ordinal/nominal predictors the regression spline model will use indicator basis functions.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca> and Zhenghua Nie <niez@mcmaster.ca>

Maintainer: Jeffrey S. Racine <racinej@mcmaster.ca>

I would like to gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada (http://www.nserc.ca), the Social Sciences and Humanities Research Council of Canada (http://www.sshrc.ca), and the Shared Hierarchical Academic Research Computing Network (http://www.sharcnet.ca).

References

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

---

clsd *Categorical Logspline Density*

Description

clsd computes the logspline density, density derivative, distribution, and smoothed quantiles for a one (1) dimensional continuous variable using the approach of Racine (2013).

Usage

```
clsd(x = NULL,
     beta = NULL,
     xeval = NULL,
     degree = NULL,
     segments = NULL,
     degree.min = 2,
     degree.max = 25,
     segments.min = 1,
     segments.max = 100,
     lbound = NULL,
     ubound = NULL,
     basis = "tensor",
     knots = "quantiles",
     penalty = NULL,
     deriv.index = 1,
```

```
deriv = 1,
elastic.max = TRUE,
elastic.diff = 3,
do.gradient = TRUE,
er = NULL,
monotone = TRUE,
monotone.lb = -250,
n.integrate = 500,
nmulti = 1,
method = c("L-BFGS-B", "Nelder-Mead", "BFGS", "CG", "SANN"),
verbose = FALSE,
quantile.seq = seq(.01,.99,by=.01),
random.seed = 42,
maxit = 10^5,
max.attempts = 25,
NOMAD = FALSE)
```

Arguments

x	a numeric vector of training data
beta	a numeric vector of coefficients (default NULL)
xeval	a numeric vector of evaluation data
degree	integer/vector specifying the polynomial degree of the B-spline basis for each dimension of the continuous x (default degree=2)
segments	integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous x (i.e. number of knots minus one) (default segments=1, i.e. Bezier curve)
segments.min,segments.max	when elastic.max=FALSE, the minimum/maximum segments of the B-spline basis for each of the continuous predictors (default segments.min=1,segments.max=100)
degree.min,degree.max	when elastic.max=FALSE the minimum/maximum degree of the B-spline basis for each of the continuous predictors (default degree.min=2, degree.max=25)
lbound,ubound	lower/upper bound for the support of the density. For example, if there is a priori knowledge that the density equals zero to the left of 0, and has a discontinuity at 0, the user could specify lbound = 0. However, if the density is essentially zero near 0, one does not need to specify lbound
basis	a character string (default basis="tensor") indicating whether the additive or tensor product B-spline basis matrix for a multivariate polynomial spline or generalized B-spline polynomial basis should be used
knots	a character string (default knots="quantiles") specifying where knots are to be placed. 'quantiles' specifies knots placed at equally spaced quantiles (equal number of observations lie in each segment) and 'uniform' specifies knots placed at equally spaced intervals
deriv	an integer l (default deriv=1) specifying whether to compute the univariate lth partial derivative for each continuous predictor (and difference in levels for each

categorical predictor) or not and if so what order. Note that if deriv is higher than the spline degree of the associated continuous predictor then the derivative will be zero and a warning issued to this effect

deriv.index — an integer l (default deriv.index=1) specifying the index (currently only supports 1) of the variable whose derivative is requested

nmulti — integer number of times to restart the process of finding extrema of the cross-validation function from different (random) initial points (default nmulti=1)

penalty — the parameter to be used in the AIC criterion. The method chooses the number of degrees plus number of segments (knots-1) that maximizes 2*logl-penalty*(degree+segments). The default is to use the penalty parameter of log(n)/2 (2 would deliver standard AIC, log(n) standard BIC)

elastic.max,elastic.diff — a logical value/integer indicating whether to use 'elastic' search bounds such that the optimal degree/segment must lie elastic.diff units from the respective search bounds

do.gradient — a logical value indicating whether or not to use the analytical gradient during optimization (defaults to TRUE)

er — a scalar indicating the fraction of data range to extend the tails (default 1/log(n), see extendrange for further details)

monotone — a logical value indicating whether modify the standard B-spline basis function so that it is tailored for density estimation (default TRUE)

monotone.lb — a negative bound specifying the lower bound on the linear segment coefficients used when (monotone=FALSE)

n.integrate — the number of evenly spaced integration points on the extended range specified by er (defaults to 500)

method — see optim for details

verbose — a logical value which when TRUE produces verbose output during optimization

quantile.seq — a sequence of numbers lying in $[0, 1]$ on which quantiles from the logspline distribution are obtained

random.seed — seeds the random number generator for initial parameter values when optim is called

maxit — maximum number of iterations used by optim

max.attempts — maximum number of attempts to undertake if optim fails for any set of initial parameters for each value of nmulti

NOMAD — a logical value which when TRUE calls snomadr to determine the optimal degree and segments

Details

Typical usages are (see below for a list of options and also the examples at the end of this help file)

```
model <- clsd(x)
```

clsd computes a logspline density estimate of a one (1) dimensional continuous variable.

The spline model employs the tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (http://www.gnu.org/software/gsl/) and the tensor.prod.model.matrix function.

When basis="additive" the model becomes additive in nature (i.e. no interaction/tensor terms thus semiparametric not fully nonparametric).

When basis="tensor" the model uses the multivariate tensor product basis.

Value

clsd returns a clsd object. The generic functions coef, fitted, plot and summary support objects of this type (er=FALSE plots the density on the sample realizations (default is 'extended range' data), see er above, distribution=TRUE plots the distribution). The returned object has the following components:

density	estimates of the density function at the sample points
density.er	the density evaluated on the 'extended range' of the data
density.deriv	estimates of the derivative of the density function at the sample points
density.deriv.er	estimates of the derivative of the density function evaluated on the 'extended range' of the data
distribution	estimates of the distribution function at the sample points
distribution.er	the distribution evaluated on the 'extended range' of the data
xer	the 'extended range' of the data
degree	integer/vector specifying the degree of the B-spline basis for each dimension of the continuous x
segments	integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous x
xq	vector of quantiles
tau	vector generated by quantile.seq or input by the user (lying in [0,1]) from which the quantiles xq are obtained

Usage Issues

This function should be considered to be in 'beta' status until further notice.

If smoother estimates are desired and degree=degree.min, increase degree.min to, say, degree.min=3.

The use of 'regression' B-splines can lead to undesirable behavior at the endpoints of the data (i.e. when monotone=FALSE). The default 'density' B-splines ought to be well-behaved in these regions.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Racine, J.S. (2013), "Logspline Mixed Data Density Estimation," manuscript.

See Also

logspline

Examples

```
## Not run:
## Old Faithful eruptions data histogram and clsd density

library(MASS)
data(faithful)
attach(faithful)

model <- clsd(eruptions)

ylim <- c(0,max(model$density,hist(eruptions,breaks=20,plot=FALSE)$density))

plot(model,ylim=ylim)

hist(eruptions,breaks=20,freq=FALSE,add=TRUE,lty=2)

rug(eruptions)

summary(model)

coef(model)

## Simulated data

set.seed(42)
require(logspline)

## Example - simulated data

n <- 250
x <- sort(rnorm(n))
f.dgp <- dnorm(x)

model <- clsd(x)

## Standard (cubic) estimate taken from the logspline package
## Compute MSEs

mse.clsd <- mean((fitted(model)-f.dgp)^2)

model.logspline <- logspline(x)

mse.logspline <- mean((dlogspline(x,model.logspline)-f.dgp)^2)
```

```
ylim <- c(0,max(fitted(model),dlogspline(x,model.logspline),f.dgp))

plot(model,
     ylim=ylim,
     sub=paste("MSE: logspline = ",format(mse.logspline),", clsd = ",
     format(mse.clsd)),
     lty=3,
     col=3)

xer <- model$xer

lines(xer,dlogspline(xer,model.logspline),col=2,lty=2)
lines(xer,dnorm(xer),col=1,lty=1)

rug(x)

legend("topright",c("DGP",
                    paste("Cubic Logspline Density (package `logspline', knots = ",
                          model.logspline$nknots,")",sep=""),
                    paste("clsd Density (degree = ", model$degree, ", segments = ",
                      model$segments,", penalty = ",round(model$penalty,2),")",sep="")),
       lty=1:3,
       col=1:3,
       bty="n",
       cex=0.75)

summary(model)

coef(model)

## Simulate data with known bounds

set.seed(42)
n <- 10000
x <- runif(n,0,1)

model <- clsd(x,lbound=0,ubound=1)

plot(model)

## End(Not run)
```

---

cps71 *Canadian High School Graduate Earnings*

Description

Canadian cross-section wage data consisting of a random sample taken from the 1971 Canadian Census Public Use Tapes for male individuals having common education (grade 13). There are 205 observations in total.

Usage

```
data("cps71")
```

Format

A data frame with 2 columns, and 205 rows.

logwage the first column, of type `numeric`

age the second column, of type `integer`

Source

Aman Ullah

References

Pagan, A. and A. Ullah (1999), *Nonparametric Econometrics,* Cambridge University Press.

Examples

```
## Example - we compare the nonparametric local linear kernel regression
## method with the regression spline for the cps71 data. Note that there
## are no categorical predictors in this dataset so we are merely
## comparing and contrasting the two nonparametric estimates.

data(cps71)
attach(cps71)
require(np)

model.crs <- crs(logwage~age,complexity="degree-knots")
model.np <- npreg(logwage~age,regtype="ll")

plot(age,logwage,cex=0.25,col="grey",
     sub=paste("crs-CV = ", formatC(model.crs$cv.score,format="f",digits=3),
       ", npreg-CV = ", formatC(model.np$bws$fval,format="f",digits=3),sep=""))
lines(age,fitted(model.crs),lty=1,col=1)
lines(age,fitted(model.np),lty=2,col=2)

crs.txt <- paste("crs (R-squared = ",formatC(model.crs$r.squared,format="f",digits=3),")",sep="")
np.txt <- paste("ll-npreg (R-squared = ",formatC(model.np$R2,format="f",digits=3),")",sep="")

legend(22.5,15,c(crs.txt,np.txt),lty=c(1,2),col=c(1,2),bty="n")

summary(model.crs)
summary(model.np)
detach("package:np")
```

crs *Categorical Regression Splines*

Description

crs computes a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and categorical (factor/ordered) predictors (Ma and Racine (2013), Ma, Racine and Yang (under revision)).

Usage

```
crs(...)
## Default S3 method:
crs(xz,
    y,
    degree = NULL,
    segments = NULL,
    include = NULL,
    kernel = TRUE,
    lambda = NULL,
    complexity = c("degree-knots","degree","knots"),
    knots = c("quantiles","uniform","auto"),
    basis = c("additive","tensor","glp","auto"),
    deriv = 0,
    data.return = FALSE,
    prune = FALSE,
    model.return = FALSE,
    tau = NULL,
    weights = NULL,
    ...)

## S3 method for class 'formula'
crs(formula,
    data = list(),
    degree = NULL,
    segments = NULL,
    include = NULL,
    degree.max = 10,
    segments.max = 10,
    degree.min = 0,
    segments.min = 1,
    cv.df.min = 1,
    cv = c("nomad","exhaustive","none"),
    cv.threshold = 1000,
    cv.func = c("cv.ls","cv.gcv","cv.aic"),
    kernel = TRUE,
    lambda = NULL,
```

```
lambda.discrete = FALSE,
lambda.discrete.num = 100,
complexity = c("degree-knots","degree","knots"),
knots = c("quantiles","uniform","auto"),
basis = c("auto","additive","tensor","glp"),
deriv = 0,
data.return = FALSE,
prune = FALSE,
model.return = FALSE,
restarts = 0,
random.seed = 42,
max.bb.eval = 10000,
initial.mesh.size.real = "r1.0e-01",
initial.mesh.size.integer = "1",
min.mesh.size.real = paste(sqrt(.Machine$double.eps)),
min.mesh.size.integer = paste(sqrt(.Machine$double.eps)),
min.poll.size.real = paste(sqrt(.Machine$double.eps)),
min.poll.size.integer = paste(sqrt(.Machine$double.eps)),
opts=list(),
nmulti = 5,
tau = NULL,
weights = NULL,
singular.ok = FALSE,
...)
```

Arguments

xz	numeric (x) and or nominal/ordinal (factor/ordered) predictors (z)
y	a numeric vector of responses.
degree	integer/vector specifying the polynomial degree of the B-spline basis for each dimension of the continuous x (default degree=3, i.e. cubic spline)
segments	integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous x (i.e. number of knots minus one) (default segments=1, i.e. Bezier curve)
include	integer/vector specifying whether each of the nominal/ordinal (factor/ordered) predictors in x are included or omitted from the resulting estimate
lambda	a vector of bandwidths for each dimension of the categorical z
lambda.discrete	if lambda.discrete=TRUE, the bandwidth will be discretized into lambda.discrete.num+1 points and lambda will be chosen from these points
lambda.discrete.num	a positive integer indicating the number of discrete values that lambda can assume - this parameter will only be used when lambda.discrete=TRUE
formula	a symbolic description of the model to be fit
data	an optional data frame containing the variables in the model

cv — a character string (default cv="nomad") indicating whether to use nonsmooth mesh adaptive direct search, exhaustive search, or no search (i.e. use user supplied values for degree, segments, and lambda)

cv.threshold — an integer (default cv.threshold=1000) for simple problems with no categorical predictors (i.e. kernel=FALSE otherwise optim/snomadr search is necessary) such that, if the number of combinations of degree/segments is less than the threshold and cv="nomad", instead use exhaustive search (cv="exhaustive")

cv.func — a character string (default cv.func="cv.ls") indicating which method to use to select smoothing parameters. cv.gcv specifies generalized cross-validation (Craven and Wahba (1979)), cv.aic specifies expected Kullback-Leibler cross-validation (Hurvich, Simonoff, and Tsai (1998)), and cv.ls specifies least-squares cross-validation

kernel — a logical value (default kernel=TRUE) indicating whether to use kernel smoothing or not

degree.max — the maximum degree of the B-spline basis for each of the continuous predictors (default degree.max=10)

segments.max — the maximum segments of the B-spline basis for each of the continuous predictors (default segments.max=10)

degree.min — the minimum degree of the B-spline basis for each of the continuous predictors (default degree.min=0)

segments.min — the minimum segments of the B-spline basis for each of the continuous predictors (default segments.min=1)

cv.df.min — the minimum degrees of freedom to allow when conducting NOMAD-based cross-validation (default cv.df.min=1)

complexity — a character string (default complexity="degree-knots") indicating whether model 'complexity' is determined by the degree of the spline or by the number of segments (i.e. number of knots minus one). This option allows the user to use cross-validation to select either the spline degree (number of knots held fixed) or the number of knots (spline degree held fixed) or both the spline degree and number of knots

For the continuous predictors the regression spline model employs either the additive or tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (http://www.gnu.org/software/gsl/) and the tensor.prod.model.matrix function

knots — a character string (default knots="quantiles") specifying where knots are to be placed. 'quantiles' specifies knots placed at equally spaced quantiles (equal number of observations lie in each segment) and 'uniform' specifies knots placed at equally spaced intervals. If knots="auto", the knot type will be automatically determined by cross-validation

basis — a character string (default basis="auto") indicating whether the additive or tensor product B-spline basis matrix for a multivariate polynomial spline or generalized B-spline polynomial basis should be used. Note this can be automatically determined by cross-validation if cv="nomad" or cv="exhaustive" and basis="auto", and is an 'all or none' proposition (i.e. interaction terms for all predictors or for no predictors given the nature of 'tensor products'). Note also

that if there is only one predictor this defaults to basis="additive" to avoid unnecessary computation as the spline bases are equivalent in this case

deriv — an integer l (default deriv=0) specifying whether to compute the univariate lth partial derivative for each continuous predictor (and difference in levels for each categorical predictor) or not and if so what order. Note that if deriv is higher than the spline degree of the associated continuous predictor then the derivative will be zero and a warning issued to this effect

data.return — a logical value indicating whether to return x,z,y or not (default data.return=FALSE)

prune — a logical value (default prune=FALSE) specifying whether the (final) model is to be 'pruned' using a stepwise cross-validation criterion based upon a modified version of stepAIC (see below for details)

model.return — a logical value indicating whether to return the list of lm models or not when kernel=TRUE (default model.return=FALSE)

restarts — integer specifying the number of times to restart the process of finding extrema of the cross-validation function (for the bandwidths only) from different (random) initial points

random.seed — when it is not missing and not equal to 0, the initial points will be generated using this seed when using frscvNOMAD or krscvNOMAD and nmulti > 0

max.bb.eval — argument passed to the NOMAD solver (see snomadr for further details)

initial.mesh.size.real — argument passed to the NOMAD solver (see snomadr for further details)

initial.mesh.size.integer — argument passed to the NOMAD solver (see snomadr for further details)

min.mesh.size.real — argument passed to the NOMAD solver (see snomadr for further details)

min.mesh.size.integer — arguments passed to the NOMAD solver (see snomadr for further details)

min.poll.size.real — arguments passed to the NOMAD solver (see snomadr for further details)

min.poll.size.integer — arguments passed to the NOMAD solver (see snomadr for further details)

opts — list of optional arguments to be passed to snomadr

nmulti — integer number of times to restart the process of finding extrema of the cross-validation function from different (random) initial points (default nmulti=5)

tau — if non-null a number in (0,1) denoting the quantile for which a quantile regression spline is to be estimated rather than estimating the conditional mean (default tau=NULL). Criterion function set by cv.func= are modified accordingly to admit quantile regression.

weights — an optional vector of weights to be used in the fitting process. Should be 'NULL' or a numeric vector. If non-NULL, weighted least squares is used with weights 'weights' (that is, minimizing 'sum(w*e^2)'); otherwise ordinary least squares is used.

singular.ok — a logical value (default singular.ok=FALSE) that, when FALSE, discards singular bases during cross-validation (a check for ill-conditioned bases is performed).

... — optional arguments

Details

Typical usages are (see below for a list of options and also the examples at the end of this help file)

```
## Estimate the model and let the basis type be determined by
## cross-validation (i.e. cross-validation will determine whether to
## use the additive, generalized, or tensor product basis)

model <- crs(y~x1+x2)

## Estimate the model for a specified degree/segment/bandwidth
## combination and do not run cross-validation (will use the
## additive basis by default)

model <- crs(y~x1+factor(x2),cv="none",degree=3,segments=1,lambda=.1)

## Plot the mean and (asymptotic) error bounds

plot(model,mean=TRUE,ci=TRUE)

## Plot the first partial derivative and (asymptotic) error bounds

plot(model,deriv=1,ci=TRUE)
```

`crs` computes a regression spline estimate of a one (1) dimensional dependent variable on an `r`-dimensional vector of continuous and categorical (`factor`/`ordered`) predictors.

The regression spline model employs the tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (`http://www.gnu.org/software/gsl/`) and the `tensor.prod.model.matrix` function.

When `basis="additive"` the model becomes additive in nature (i.e. no interaction/tensor terms thus semiparametric not fully nonparametric).

When `basis="tensor"` the model uses the multivariate tensor product basis.

When `kernel=FALSE` the model uses indicator basis functions for the nominal/ordinal (`factor`/`ordered`) predictors rather than kernel weighting.

When `kernel=TRUE` the product kernel function for the discrete predictors is of the 'Li-Racine' type (see Li and Racine (2007) for details).

When `cv="nomad"`, numerical search is undertaken using Nonsmooth Optimization by Mesh Adaptive Direct Search (Abramson, Audet, Couture, Dennis, Jr., and Le Digabel (2011)).

When `kernel=TRUE` and `cv="exhaustive"`, numerical search is undertaken using `optim` and the box-constrained `L-BFGS-B` method (see `optim` for details). The user may restart the algorithm as many times as desired via the `restarts` argument (default `restarts=0`). The approach ascends from `degree=0` (or `segments=0`) through `degree.max` and for each value of `degree` (or `segments`) searches for the optimal bandwidths. After the most complex model has been searched then the optimal `degree`/`segments`/`lambda` combination is selected. If any element of the optimal `degree` (or `segments`) vector coincides with `degree.max` (or `segments.max`) a warning is produced and the user ought to restart their search with a larger value of `degree.max` (or `segments.max`).

Note that the default plot method for a crs object provides some diagnostic measures, in particular, a) residuals versus fitted values (useful for checking the assumption that E(u|x)=0), b) a normal quantile-quantile plot which allows residuals to be assessed for normality (qqnorm), c) a scale-location plot that is useful for checking the assumption that the errors are iid and, in particular, that the variance is homogeneous, and d) 'Cook's distance' which computes the single-case influence function. See below for other arguments for the plot function for a crs object.

Note that setting prune=TRUE produces a final 'pruning' of the model via a stepwise cross-validation criterion achieved by modifying stepAIC and replacing extractAIC with extractCV throughout the function. This option may be enabled to remove potentially superfluous bases thereby improving the finite-sample efficiency of the resulting model. Note that if the cross-validation score for the pruned model is no better than that for the original model then the original model is returned with a warning to this effect. Note also that this option can only be used when kernel=FALSE.

Value

crs returns a crs object. The generic functions fitted and residuals extract (or generate) estimated values and residuals. Furthermore, the functions summary, predict, and plot (options mean=FALSE, deriv=i where i is an integer, ci=FALSE, persp.rgl=FALSE, plot.behavior=c("plot","plot-data","data
xtrim=0.0,xq=0.5) support objects of this type. The returned object has the following components:

fitted.values	estimates of the regression function (conditional mean) at the sample points or evaluation points
lwr,upr	lower/upper bound for a 95% confidence interval for the fitted.values (conditional mean) obtained from predict.lm via the argument interval="confidence"
residuals	residuals computed at the sample points or evaluation points
degree	integer/vector specifying the degree of the B-spline basis for each dimension of the continuous x
segments	integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous x
include	integer/vector specifying whether each of the nominal/ordinal (factor/ordered) predictors z are included or omitted from the resulting estimate if kernel=FALSE (see below)
kernel	a logical value indicating whether kernel smoothing was used (kernel=TRUE) or not
lambda	vector of bandwidths used if kernel=TRUE
call	a symbolic description of the model
r.squared	coefficient of determination (Doksum and Samarov (1995))
model.lm	an object of 'class' 'lm' if kernel=FALSE or a list of objects of 'class' 'lm' if kernel=TRUE (accessed by model.lm[[1]], model.lm[[2]],...,. By way of example, if foo is a crs object and kernel=FALSE, then foo$model.lm is an object of 'class' 'lm', while objects of 'class' 'lm' return the model.frame in model.lm$model which can be accessed via foo$model.lm$model where foo is the crs object (the model frame foo$model.lm$model contains the B-spline bases underlying the estimate which might be of interest). Again by way of example, when kernel=TRUE then foo$model.lm[[1]]$model contains the model frame for the first unique combination of categorical predictors,

	foo$model.lm[[2]]$model the second and so forth (the weights will potentially differ for each model depending on the value(s) of lambda)
deriv.mat	a matrix of derivatives (or differences in levels for the categorical z) whose order is determined by deriv= in the crs call
deriv.mat.lwr	a matrix of 95% coverage lower bounds for deriv.mat
deriv.mat.upr	a matrix of 95% coverage upper bounds for deriv.mat
hatvalues	the hatvalues for the estimated model
P.hat	the kernel probability estimates corresponding to the categorical predictors in the estimated model

Usage Issues

Note that when kernel=FALSE summary supports the option sigtest=TRUE that conducts an F-test for significance for each predictor.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Abramson, M.A. and C. Audet and G. Couture and J.E. Dennis Jr. and and S. Le Digabel (2011), "The NOMAD project". Software available at http://www.gerad.ca/nomad.

Craven, P. and G. Wahba (1979), "Smoothing Noisy Data With Spline Functions," Numerische Mathematik, 13, 377-403.

Doksum, K. and A. Samarov (1995), "Nonparametric Estimation of Global Functionals and a Measure of the Explanatory Power of Covariates in Regression," The Annals of Statistics, 23 1443-1473.

Hurvich, C.M. and J.S. Simonoff and C.L. Tsai (1998), "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion," Journal of the Royal Statistical Society B, 60, 271-293.

Le Digabel, S. (2011), "Algorithm 909: NOMAD: Nonlinear Optimization With The MADS Algorithm". ACM Transactions on Mathematical Software, 37(4):44:1-44:15.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

Racine, J.S. (2011), "Cross-Validated Quantile Regression Splines," manuscript.

See Also

smooth.spline, loess, npreg

Examples

```
set.seed(42)
## Example - simulated data
n <- 1000
num.eval <- 50
x1 <- runif(n)
x2 <- runif(n)
z <- rbinom(n,1,.5)
dgp <- cos(2*pi*x1)+sin(2*pi*x2)+z
z <- factor(z)
y <- dgp + rnorm(n,sd=.5)

## Estimate a model with specified degree, segments, and bandwidth
model <- crs(y~x1+x2+z,degree=c(5,5),
                       segments=c(1,1),
                       lambda=0.1,
                       cv="none",
                       kernel=TRUE)
summary(model)

## Perspective plot
x1.seq <- seq(min(x1),max(x1),length=num.eval)
x2.seq <- seq(min(x2),max(x2),length=num.eval)
x.grid <- expand.grid(x1.seq,x2.seq)
newdata <- data.frame(x1=x.grid[,1],x2=x.grid[,2],
                      z=factor(rep(0,num.eval**2),levels=c(0,1)))
z0 <- matrix(predict(model,newdata=newdata),num.eval,num.eval)
newdata <- data.frame(x1=x.grid[,1],x2=x.grid[,2],
                      z=factor(rep(1,num.eval),levels=c(0,1)))
z1 <- matrix(predict(model,newdata=newdata),num.eval,num.eval)
zlim=c(min(z0,z1),max(z0,z1))
persp(x=x1.seq,y=x2.seq,z=z0,
      xlab="x1",ylab="x2",zlab="y",zlim=zlim,
      ticktype="detailed",
      border="red",
      theta=45,phi=45)
par(new=TRUE)
persp(x=x1.seq,y=x2.seq,z=z1,
      xlab="x1",ylab="x2",zlab="y",zlim=zlim,
      theta=45,phi=45,
      ticktype="detailed",
      border="blue")

## Partial regression surface plot
plot(model,mean=TRUE,ci=TRUE)
## Not run:
## A plot example where we extract the partial surfaces, confidence
## intervals etc. automatically generated by plot(mean=TRUE,...) but do
## not plot, rather save for separate use.
pdat <- plot(model,mean=TRUE,ci=TRUE,plot.behavior="data")

## Column 1 is the (evaluation) predictor ([,1]), 2-4 ([,-1]) the mean,
```

```
## lwr, and upr (note the returned value is a `list' hence pdat[[1]] is
## data for the first predictor, pdat[[2]] the second etc). Note that
## matplot() can plot this nicely.
matplot(pdat[[1]][,1],pdat[[1]][,-1],
        xlab=names(pdat[[1]][1]),ylab=names(pdat[[1]][2]),
        lty=c(1,2,2),col=c(1,2,2),type="l")

## End(Not run)
```

---

crsiv *Nonparametric Instrumental Regression*

Description

crsiv computes nonparametric estimation of an instrumental regression function φ defined by conditional moment restrictions stemming from a structural econometric model: $E[Y-\varphi(Z,X)|W]=0$, and involving endogenous variables Y and Z, exogenous variables X, and instruments W. The function φ is the solution of an ill-posed inverse problem.

When method="Tikhonov", crsiv uses the approach of Darolles, Fan, Florens and Renault (2011) modified for regression splines (Darolles et al use local constant kernel weighting). When method="Landweber-Fridman", crsiv uses the approach of Horowitz (2011) using the regression spline methodology implemented in the **crs** package.

Usage

```
crsiv(y,
      z,
      w,
      x = NULL,
      zeval = NULL,
      weval = NULL,
      xeval = NULL,
      alpha = NULL,
      alpha.min = 1e-10,
      alpha.max = 1e-01,
      alpha.tol = .Machine$double.eps^0.25,
      deriv = 0,
      iterate.max = 1000,
      iterate.diff.tol = 1.0e-08,
      constant = 0.5,
      penalize.iteration = TRUE,
      smooth.residuals = TRUE,
      start.from = c("Eyz","EEywz"),
      starting.values = NULL,
      stop.on.increase = TRUE,
      method = c("Landweber-Fridman","Tikhonov"),
      opts = list("MAX_BB_EVAL"=10000,
```

```
                    "EPSILON"=.Machine$double.eps,
                    "INITIAL_MESH_SIZE"="r1.0e-01",
                    "MIN_MESH_SIZE"=paste("r",sqrt(.Machine$double.eps),sep=""),
                    "MIN_POLL_SIZE"=paste("r",sqrt(.Machine$double.eps),sep=""),
                    "DISPLAY_DEGREE"=0),
        ...)
```

Arguments

y	a one (1) dimensional numeric or integer vector of dependent data, each element i corresponding to each observation (row) i of z
z	a p-variate data frame of endogenous predictors. The data types may be continuous, discrete (unordered and ordered factors), or some combination thereof
w	a q-variate data frame of instruments. The data types may be continuous, discrete (unordered and ordered factors), or some combination thereof
x	an r-variate data frame of exogenous predictors. The data types may be continuous, discrete (unordered and ordered factors), or some combination thereof
zeval	a p-variate data frame of endogenous predictors on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by z
weval	a q-variate data frame of instruments on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by w
xeval	an r-variate data frame of exogenous predictors on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by x
alpha	a numeric scalar that, if supplied, is used rather than numerically solving for alpha, when using method="Tikhonov"
alpha.min	minimum of search range for α, the Tikhonov regularization parameter, when using method="Tikhonov"
alpha.max	maximum of search range for α, the Tikhonov regularization parameter, when using method="Tikhonov"
alpha.tol	the search tolerance for optimize when solving for α, the Tikhonov regularization parameter, when using method="Tikhonov"
iterate.max	an integer indicating the maximum number of iterations permitted before termination occurs when using method="Landweber-Fridman"
iterate.diff.tol	the search tolerance for the difference in the stopping rule from iteration to iteration when using method="Landweber-Fridman" (disable by setting to zero)
constant	the constant to use when using method="Landweber-Fridman"
method	the regularization method employed (default "Landweber-Fridman", see Horowitz (2011); see Darolles, Fan, Florens and Renault (2011) for details for "Tikhonov")
penalize.iteration	a logical value indicating whether to penalize the norm by the number of iterations or not (default TRUE)

`smooth.residuals`
: a logical value (defaults to `TRUE`) indicating whether to optimize bandwidths for the regression of $y - \varphi(z)$ on w or for the regression of $\varphi(z)$ on w during iteration

`start.from`
: a character string indicating whether to start from $E(Y|z)$ (default, `"Eyz"`) or from $E(E(Y|z)|z)$ (this can be overridden by providing `starting.values` below)

`starting.values`
: a value indicating whether to commence Landweber-Fridman assuming $\varphi_{-1} = starting.values$ (proper Landweber-Fridman) or instead begin from $E(y|z)$ (defaults to `NULL`, see details below)

`stop.on.increase`
: a logical value (defaults to `TRUE`) indicating whether to halt iteration if the stopping criterion (see below) increases over the course of one iteration (i.e. it may be above the iteration tolerance but increased)

`opts`
: arguments passed to the NOMAD solver (see `snomadr` for further details)

`deriv`
: an integer `l` (default `deriv=0`) specifying whether to compute the univariate `l`th partial derivative for each continuous predictor (and difference in levels for each categorical predictor) or not and if so what order. Note that if `deriv` is higher than the spline degree of the associated continuous predictor then the derivative will be zero and a warning issued to this effect (see important note below)

`...`
: additional arguments supplied to `crs`

Details

Tikhonov regularization requires computation of weight matrices of dimension $n \times n$ which can be computationally costly in terms of memory requirements and may be unsuitable (i.e. unfeasible) for large datasets. Landweber-Fridman will be preferred in such settings as it does not require construction and storage of these weight matrices while it also avoids the need for numerical optimization methods to determine α, though it does require iteration that may be equally or even more computationally demanding in terms of total computation time.

When using `method="Landweber-Fridman"`, an optimal stopping rule based upon $||E(y|w) - E(\varphi_k(z,x)|w)||^2$ is used to terminate iteration. However, if local rather than global optima are encountered the resulting estimates can be overly noisy. To best guard against this eventuality set `nmulti` to a larger number than the default `nmulti=5` for `crs` when using `cv="nomad"` or instead use `cv="exhaustive"` if possible (this may not be feasible for non-trivial problems).

When using `method="Landweber-Fridman"`, iteration will terminate when either the change in the value of $||(E(y|w) - E(\varphi_k(z,x)|w))/E(y|w)||^2$ from iteration to iteration is less than `iterate.diff.tol` or we hit `iterate.max` or $||(E(y|w) - E(\varphi_k(z,x)|w))/E(y|w)||^2$ stops falling in value and starts rising.

When your problem is a simple one (e.g. univariate Z, W, and X) you might want to avoid `cv="nomad"` and instead use `cv="exhaustive"` since exhaustive search may be feasible (for `degree.max` and `segments.max` not overly large). This will guarantee an exact solution for each iteration (i.e. there will be no errors arising due to numerical search).

`demo(crsiv)`, `demo(crsiv_exog)`, and `demo(crsiv_exog_persp)` provide flexible interactive demonstrations similar to the example below that allow you to modify and experiment with parameters such as the sample size, method, and so forth in an interactive session.

Value

crsiv returns a crs object. The generic functions fitted and residuals extract (or generate) estimated values and residuals. Furthermore, the functions summary, predict, and plot (options mean=FALSE, deriv=i where i is an integer, ci=FALSE, plot.behavior=c("plot","plot-data","data")) support objects of this type.

See crs for details on the return object components.

In addition to the standard crs components, crsiv returns components phi and either alpha when method="Tikhonov" or phi, phi.mat, num.iterations, norm.stop, norm.value and convergence when method="Landweber-Fridman".

Note

Using the option deriv= computes (effectively) the analytical derivative of the estimated $\varphi(Z, X)$ and not that using crsivderiv, which instead uses the method of Florens and Racine (2012). Though both are statistically consistent, practitioners may desire one over the other hence we provide both.

Note

This function should be considered to be in 'beta test' status until further notice.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>, Samuele Centorrino <samuele.centorrino@univ-tlse1.fr>

References

Carrasco, M. and J.P. Florens and E. Renault (2007), "Linear Inverse Problems in Structural Econometrics Estimation Based on Spectral Decomposition and Regularization," In: James J. Heckman and Edward E. Leamer, Editor(s), Handbook of Econometrics, Elsevier, 2007, Volume 6, Part 2, Chapter 77, Pages 5633-5751

Darolles, S. and Y. Fan and J.P. Florens and E. Renault (2011), "Nonparametric Instrumental Regression," Econometrica, 79, 1541-1565.

Feve, F. and J.P. Florens (2010), "The Practice of Non-parametric Estimation by Solving Inverse Problems: The Example of Transformation Models," Econometrics Journal, 13, S1-S27.

Florens, J.P. and J.S. Racine (2012), "Nonparametric Instrumental Derivatives," Working Paper.

Fridman, V. M. (1956), "A Method of Successive Approximations for Fredholm Integral Equations of the First Kind," Uspeskhi, Math. Nauk., 11, 233-334, in Russian.

Horowitz, J.L. (2011), "Applied Nonparametric Instrumental Variables Estimation," Econometrica, 79, 347-394.

Landweber, L. (1951), "An Iterative Formula for Fredholm Integral Equations of the First Kind," American Journal of Mathematics, 73, 615-24.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

See Also

npreg, crs

Examples

```
## Not run:
## This illustration was made possible by Samuele Centorrino
## <samuele.centorrino@univ-tlse1.fr>

set.seed(42)
n <- 1500

## The DGP is as follows:

## 1) y = phi(z) + u

## 2) E(u|z) != 0 (endogeneity present)

## 3) Suppose there exists an instrument w such that z = f(w) + v and
## E(u|w) = 0

## 4) We generate v, w, and generate u such that u and z are
## correlated. To achieve this we express u as a function of v (i.e. u =
## gamma v + eps)

v <- rnorm(n,mean=0,sd=0.27)
eps <- rnorm(n,mean=0,sd=0.05)
u <- -0.5*v + eps
w <- rnorm(n,mean=0,sd=1)

## In Darolles et al (2011) there exist two DGPs. The first is
## phi(z)=z^2 and the second is phi(z)=exp(-abs(z)) (which is
## discontinuous and has a kink at zero).

fun1 <- function(z) { z^2 }
fun2 <- function(z) { exp(-abs(z)) }

z <- 0.2*w + v

## Generate two y vectors for each function.

y1 <- fun1(z) + u
y2 <- fun2(z) + u

## You set y to be either y1 or y2 (ditto for phi) depending on which
## DGP you are considering:

y <- y1
phi <- fun1

## Create an evaluation dataset sorting on z (for plotting)
```

```
evaldata <- data.frame(y,z,w)
evaldata <- evaldata[order(evaldata$z),]

## Compute the non-IV regression spline estimator of E(y|z)

model.noniv <- crs(y~z,opts=opts)
mean.noniv <- predict(model.noniv,newdata=evaldata)

## Compute the IV-regression spline estimator of phi(z)

model.iv <- crsiv(y=y,z=z,w=w)
phi.iv <- predict(model.iv,newdata=evaldata)

## For the plots, restrict focal attention to the bulk of the data
## (i.e. for the plotting area trim out 1/4 of one percent from each
## tail of y and z)

trim <- 0.0025

curve(phi,min(z),max(z),
      xlim=quantile(z,c(trim,1-trim)),
      ylim=quantile(y,c(trim,1-trim)),
      ylab="Y",
      xlab="Z",
      main="Nonparametric Instrumental Spline Regression",
      sub=paste("Landweber-Fridman: iterations = ", model.iv$num.iterations,sep=""),
      lwd=1,lty=1)

points(z,y,type="p",cex=.25,col="grey")

lines(evaldata$z,evaldata$z^2 -0.325*evaldata$z,lwd=1,lty=1)

lines(evaldata$z,phi.iv,col="blue",lwd=2,lty=2)

lines(evaldata$z,mean.noniv,col="red",lwd=2,lty=4)

legend(quantile(z,trim),quantile(y,1-trim),
       c(expression(paste(varphi(z),", E(y|z)",sep="")),
         expression(paste("Nonparametric ",hat(varphi)(z))),
         "Nonparametric E(y|z)"),
       lty=c(1,2,4),
       col=c("black","blue","red"),
       lwd=c(1,2,2))

## End(Not run)
```

---

crsivderiv *Nonparametric Instrumental Derivatives*

Description

crsivderiv uses the approach of Florens and Racine (2012) to compute the partial derivative of a nonparametric estimation of an instrumental regression function φ defined by conditional moment restrictions stemming from a structural econometric model: $E[Y - \varphi(Z, X)|W] = 0$, and involving endogenous variables Y and Z and exogenous variables X and instruments W. The derivative function φ' is the solution of an ill-posed inverse problem, and is computed using Landweber-Fridman regularization.

Usage

```
crsivderiv(y,
           z,
           w,
           x = NULL,
           zeval = NULL,
           weval = NULL,
           xeval = NULL,
           iterate.max = 1000,
           iterate.diff.tol = 1.0e-08,
           constant = 0.5,
           penalize.iteration = TRUE,
           start.from = c("Eyz","EEywz"),
           starting.values = NULL,
           stop.on.increase = TRUE,
           smooth.residuals = TRUE,
           opts = list("MAX_BB_EVAL"=10000,
                       "EPSILON"=.Machine$double.eps,
                       "INITIAL_MESH_SIZE"="r1.0e-01",
                   "MIN_MESH_SIZE"=paste("r",sqrt(.Machine$double.eps),sep=""),
                   "MIN_POLL_SIZE"=paste("r",sqrt(.Machine$double.eps),sep=""),
                       "DISPLAY_DEGREE"=0),
           ...)
```

Arguments

y	a one (1) dimensional numeric or integer vector of dependent data, each element i corresponding to each observation (row) i of z
z	a p-variate data frame of endogenous predictors. The data types may be continuous, discrete (unordered and ordered factors), or some combination thereof
w	a q-variate data frame of instruments. The data types may be continuous, discrete (unordered and ordered factors), or some combination thereof
x	an r-variate data frame of exogenous predictors. The data types may be continuous, discrete (unordered and ordered factors), or some combination thereof
zeval	a p-variate data frame of endogenous predictors on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by z

`weval` a q-variate data frame of instruments on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by `w`

`xeval` an r-variate data frame of exogenous predictors on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by `x`

`iterate.max` an integer indicating the maximum number of iterations permitted before termination occurs when using Landweber-Fridman iteration

`iterate.diff.tol` the search tolerance for the difference in the stopping rule from iteration to iteration when using Landweber-Fridman (disable by setting to zero)

`constant` the constant to use when using Landweber-Fridman iteration

`penalize.iteration` a logical value indicating whether to penalize the norm by the number of iterations or not (default `TRUE`)

`start.from` a character string indicating whether to start from $E(Y|z)$ (default, `"Eyz"`) or from $E(E(Y|z)|z)$ (this can be overridden by providing `starting.values` below)

`starting.values` a value indicating whether to commence Landweber-Fridman assuming $\varphi'_{-1} = starting.values$ (proper Landweber-Fridman) or instead begin from $E(y|z)$ (defaults to `NULL`, see details below)

`stop.on.increase` a logical value (defaults to `TRUE`) indicating whether to halt iteration if the stopping criterion (see below) increases over the course of one iteration (i.e. it may be above the iteration tolerance but increased)

`smooth.residuals` a logical value (defaults to `TRUE`) indicating whether to optimize bandwidths for the regression of $y - \varphi(z)$ on w or for the regression of $\varphi(z)$ on w during iteration

`opts` arguments passed to the NOMAD solver (see `snomadr` for further details)

`...` additional arguments supplied to `crs`

Details

For Landweber-Fridman iteration, an optimal stopping rule based upon $||E(y|w)-E(\varphi_k(z,x)|w)||^2$ is used to terminate iteration. However, if local rather than global optima are encountered the resulting estimates can be overly noisy. To best guard against this eventuality set `nmulti` to a larger number than the default `nmulti=5` for `crs` when using `cv="nomad"` or instead use `cv="exhaustive"` if possible (this may not be feasible for non-trivial problems).

When using Landweber-Fridman iteration, iteration will terminate when either the change in the value of $||(E(y|w)-E(\varphi_k(z,x)|w))/E(y|w)||^2$ from iteration to iteration is less than `iterate.diff.tol` or we hit `iterate.max` or $||(E(y|w) - E(\varphi_k(z,x)|w))/E(y|w)||^2$ stops falling in value and starts rising.

When your problem is a simple one (e.g. univariate Z, W, and X) you might want to avoid `cv="nomad"` and instead use `cv="exhaustive"` since exhaustive search may be feasible (for `degree.max` and `segments.max` not overly large). This will guarantee an exact solution for each iteration (i.e. there will be no errors arising due to numerical search).

Value

crsivderiv returns components phi.prime, phi, phi.prime.mat, num.iterations, norm.stop, norm.value and convergence.

Note

This function currently supports univariate z only. This function should be considered to be in 'beta test' status until further notice.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Carrasco, M. and J.P. Florens and E. Renault (2007), "Linear Inverse Problems in Structural Econometrics Estimation Based on Spectral Decomposition and Regularization," In: James J. Heckman and Edward E. Leamer, Editor(s), Handbook of Econometrics, Elsevier, 2007, Volume 6, Part 2, Chapter 77, Pages 5633-5751

Darolles, S. and Y. Fan and J.P. Florens and E. Renault (2011), "Nonparametric Instrumental Regression," Econometrica, 79, 1541-1565.

Feve, F. and J.P. Florens (2010), "The Practice of Non-parametric Estimation by Solving Inverse Problems: The Example of Transformation Models," Econometrics Journal, 13, S1-S27.

Florens, J.P. and J.S. Racine (2012), "Nonparametric Instrumental Derivatives," Working Paper.

Fridman, V. M. (1956), "A Method of Successive Approximations for Fredholm Integral Equations of the First Kind," Uspeskhi, Math. Nauk., 11, 233-334, in Russian.

Horowitz, J.L. (2011), "Applied Nonparametric Instrumental Variables Estimation," Econometrica, 79, 347-394.

Landweber, L. (1951), "An Iterative Formula for Fredholm Integral Equations of the First Kind," American Journal of Mathematics, 73, 615-24.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

See Also

npreg, crsiv, crs

Examples

```
## Not run:
## This illustration was made possible by Samuele Centorrino
## <samuele.centorrino@univ-tlse1.fr>

set.seed(42)
n <- 1000

## For trimming the plot (trim .5% from each tail)
```

```
trim <- 0.005

## The DGP is as follows:

## 1) y = phi(z) + u

## 2) E(u|z) != 0 (endogeneity present)

## 3) Suppose there exists an instrument w such that z = f(w) + v and
## E(u|w) = 0

## 4) We generate v, w, and generate u such that u and z are
## correlated. To achieve this we express u as a function of v (i.e. u =
## gamma v + eps)

v <- rnorm(n,mean=0,sd=0.27)
eps <- rnorm(n,mean=0,sd=0.05)
u <- -0.5*v + eps
w <- rnorm(n,mean=0,sd=1)

## In Darolles et al (2011) there exist two DGPs. The first is
## phi(z)=z^2 and the second is phi(z)=exp(-abs(z)) (which is
## discontinuous and has a kink at zero).

fun1 <- function(z) { z^2 }
fun2 <- function(z) { exp(-abs(z)) }

z <- 0.2*w + v

## Generate two y vectors for each function.

y1 <- fun1(z) + u
y2 <- fun2(z) + u

## You set y to be either y1 or y2 (ditto for phi) depending on which
## DGP you are considering:

y <- y1
phi <- fun1

## Sort on z (for plotting)

ivdata <- data.frame(y,z,w,u,v)
ivdata <- ivdata[order(ivdata$z),]
rm(y,z,w,u,v)
attach(ivdata)

model.ivderiv <- crsivderiv(y=y,z=z,w=w)

ylim <-c(quantile(model.ivderiv$phi.prime,trim),
         quantile(model.ivderiv$phi.prime,1-trim))
```

```
plot(z,model.ivderiv$phi.prime,
     xlim=quantile(z,c(trim,1-trim)),
     main="",
     ylim=ylim,
     xlab="Z",
     ylab="Derivative",
     type="l",
     lwd=2)
rug(z)

## End(Not run)
```

---

crssigtest *Regression Spline Significance Test with Mixed Data Types*

Description

crssigtest implements a consistent test of significance of an explanatory variable in a nonparametric regression setting that is analogous to a simple t-test in a parametric regression setting. The test is based on Ma and Racine (2011).

Usage

```
crssigtest(model = NULL,
           index = NULL,
           boot.num = 399,
           boot.type = c("residual","reorder"),
           random.seed = 42,
           boot = TRUE)
```

Arguments

model	a crs model object.
index	a vector of indices for the columns of model$xz for which the test of significance is to be conducted. Defaults to (1,2,...,p) where p is the number of columns in model$xz.
boot.num	an integer value specifying the number of bootstrap replications to use. Defaults to 399.
boot.type	whether to conduct 'residual' bootstrapping (iid) or permute (reorder) in place the predictor being tested when imposing the null.
random.seed	an integer used to seed R's random number generator. This is to ensure replicability. Defaults to 42.
boot	a logical value (default TRUE) indicating whether to compute the bootstrap P-value or simply return the asymptotic P-value.

Value

crssigtest returns an object of type sigtest. summary supports sigtest objects. It has the following components:

index	the vector of indices input
P	the vector of bootstrap P-values for each statistic in F
P.asy	the vector of asymptotic P-values for each statistic in index
F	the vector of pseudo F-statistics F
F.boot	the matrix of bootstrapped pseudo F-statistics generated under the null (one column for each statistic in F)
df1	the vector of numerator degrees of freedom for each statistic in F (based on the smoother matrix)
df2	the vector of denominator degrees of freedom for each statistic in F (based on the smoother matrix)
rss	the vector of restricted sums of squared residuals for each statistic in F
uss	the vector of unrestricted sums of squared residuals for each statistic in F
boot.num	the number of bootstrap replications
boot.type	the boot.type
xnames	the names of the variables in model$xz

Usage Issues

This function should be considered to be in 'beta status' until further notice.

Caution: bootstrap methods are, by their nature, *computationally intensive*. This can be frustrating for users possessing large datasets. For exploratory purposes, you may wish to override the default number of bootstrap replications, say, setting them to boot.num=99.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine, (2011), "Inference for Regression Splines with Categorical and Continuous Predictors," Working Paper.

Examples

```
## Not run:
options(crs.messages=FALSE)
set.seed(42)

n <- 1000
```

```
z <- rbinom(n,1,.5)
x1 <- rnorm(n)
x2 <- runif(n,-2,2)
z <- factor(z)
## z is irrelevant
y <- x1 + x2 + rnorm(n)

model <- crs(y~x1+x2+z,complexity="degree",segments=c(1,1))
summary(model)

model.sigtest <- crssigtest(model)
summary(model.sigtest)

## End(Not run)
```

`Engel95` *1995 British Family Expenditure Survey*

Description

British cross-section data consisting of a random sample taken from the British Family Expenditure Survey for 1995. The households consist of married couples with an employed head-of-household between the ages of 25 and 55 years. There are 1655 household-level observations in total.

Usage

```
data("Engel95")
```

Format

A data frame with 10 columns, and 1655 rows.

food expenditure share on food, of type `numeric`

catering expenditure share on catering, of type `numeric`

alcohol expenditure share on alcohol, of type `numeric`

fuel expenditure share on fuel, of type `numeric`

motor expenditure share on motor, of type `numeric`

fares expenditure share on fares, of type `numeric`

leisure expenditure share on leisure, of type `numeric`

logexp logarithm of total expenditure, of type `numeric`

logwages logarithm of total earnings, of type `numeric`

nkids number of children, of type `numeric`

Source

Richard Blundell and Dennis Kristensen

References

Blundell, R. and X. Chen and D. Kristensen (2007), "Semi-Nonparametric IV Estimation of Shape-Invariant Engel Curves," Econometrica, 75, 1613-1669.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Examples

```
## Not run:
## Example - we compute nonparametric instrumental regression of an
## Engel curve for food expenditure shares using Landweber-Fridman
## iteration of Fredholm integral equations of the first kind.

## We consider an equation with an endogenous predictor (`z') and an
## instrument (`w'). Let y = phi(z) + u where phi(z) is the function of
## interest. Here E(u|z) is not zero hence the conditional mean E(y|z)
## does not coincide with the function of interest, but if there exists
## an instrument w such that E(u|w) = 0, then we can recover the
## function of interest by solving an ill-posed inverse problem.

data(Engel95)

## Sort on logexp (the endogenous predictor) for plotting purposes
## (i.e. so we can plot a curve for the fitted values versus logexp)

Engel95 <- Engel95[order(Engel95$logexp),]

attach(Engel95)

model.iv <- crsiv(y=food,z=logexp,w=logwages,method="Landweber-Fridman")
phihat <- model.iv$phi

## Compute the non-IV regression (i.e. regress y on z)

ghat <- crs(food~logexp)

## For the plots, we restrict focal attention to the bulk of the data
## (i.e. for the plotting area trim out 1/4 of one percent from each
## tail of y and z). This is often helpful as estimates in the tails of
## the support are less reliable (i.e. more variable) so we are
## interested in examining the relationship `where the action is'.

trim <- 0.0025

plot(logexp,food,
     ylab="Food Budget Share",
     xlab="log(Total Expenditure)",
     xlim=quantile(logexp,c(trim,1-trim)),
     ylim=quantile(food,c(trim,1-trim)),
     main="Nonparametric Instrumental Regression Splines",
     type="p",
```

```
     cex=.5,
     col="lightgrey")

lines(logexp,phihat,col="blue",lwd=2,lty=2)

lines(logexp,fitted(ghat),col="red",lwd=2,lty=4)

legend(quantile(logexp,trim),quantile(food,1-trim),
       c(expression(paste("Nonparametric IV: ",hat(varphi)(logexp))),
         "Nonparametric Regression: E(food | logexp)"),
       lty=c(2,4),
       col=c("blue","red"),
       lwd=c(2,2),
       bty="n")

## End(Not run)
```

---

frscv *Categorical Factor Regression Spline Cross-Validation*

Description

frscv computes exhaustive cross-validation directed search for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous predictors and nominal/ordinal (factor/ordered) predictors.

Usage

```
frscv(xz,
      y,
      degree.max = 10,
      segments.max = 10,
      degree.min = 0,
      segments.min = 1,
      complexity = c("degree-knots","degree","knots"),
      knots = c("quantiles","uniform","auto"),
      basis = c("additive","tensor","glp","auto"),
      cv.func = c("cv.ls","cv.gcv","cv.aic"),
      degree = degree,
      segments = segments,
      tau = NULL,
      weights = NULL,
      singular.ok = FALSE)
```

Arguments

y	continuous univariate vector
xz	continuous and/or nominal/ordinal (factor/ordered) predictors

`degree.max` the maximum degree of the B-spline basis for each of the continuous predictors (default `degree.max=10`)

`segments.max` the maximum segments of the B-spline basis for each of the continuous predictors (default `segments.max=10`)

`degree.min` the minimum degree of the B-spline basis for each of the continuous predictors (default `degree.min=0`)

`segments.min` the minimum segments of the B-spline basis for each of the continuous predictors (default `segments.min=1`)

`complexity` a character string (default `complexity="degree-knots"`) indicating whether model 'complexity' is determined by the degree of the spline or by the number of segments ('knots'). This option allows the user to use cross-validation to select either the spline degree (number of knots held fixed) or the number of knots (spline degree held fixed) or both the spline degree and number of knots

`knots` a character string (default `knots="quantiles"`) specifying where knots are to be placed. 'quantiles' specifies knots placed at equally spaced quantiles (equal number of observations lie in each segment) and 'uniform' specifies knots placed at equally spaced intervals. If `knots="auto"`, the knot type will be automatically determined by cross-validation

`basis` a character string (default `basis="additive"`) indicating whether the additive or tensor product B-spline basis matrix for a multivariate polynomial spline or generalized B-spline polynomial basis should be used. Note this can be automatically determined by cross-validation if `cv=TRUE` and `basis="auto"`, and is an 'all or none' proposition (i.e. interaction terms for all predictors or for no predictors given the nature of 'tensor products'). Note also that if there is only one predictor this defaults to `basis="additive"` to avoid unnecessary computation as the spline bases are equivalent in this case

`cv.func` a character string (default `cv.func="cv.ls"`) indicating which method to use to select smoothing parameters. `cv.gcv` specifies generalized cross-validation (Craven and Wahba (1979)), `cv.aic` specifies expected Kullback-Leibler cross-validation (Hurvich, Simonoff, and Tsai (1998)), and `cv.ls` specifies least-squares cross-validation

`degree` integer/vector specifying the degree of the B-spline basis for each dimension of the continuous `x`

`segments` integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous `x` (i.e. number of knots minus one)

`tau` if non-null a number in (0,1) denoting the quantile for which a quantile regression spline is to be estimated rather than estimating the conditional mean (default `tau=NULL`)

`weights` an optional vector of weights to be used in the fitting process. Should be 'NULL' or a numeric vector. If non-NULL, weighted least squares is used with weights 'weights' (that is, minimizing 'sum(w*e^2)'); otherwise ordinary least squares is used.

`singular.ok` a logical value (default `singular.ok=FALSE`) that, when `FALSE`, discards singular bases during cross-validation (a check for ill-conditioned bases is performed).

Details

frscv computes exhaustive cross-validation for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and nominal/ordinal (factor/ordered) predictors. The optimal K/I combination (i.e.\ degree/segments/I) is returned along with other results (see below for return values).

For the continuous predictors the regression spline model employs either the additive or tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (http://www.gnu.org/software/gsl/) and the tensor.prod.model.matrix function.

For the nominal/ordinal (factor/ordered) predictors the regression spline model uses indicator basis functions.

Value

frscv returns a crscv object. Furthermore, the function summary supports objects of this type. The returned objects have the following components:

K	scalar/vector containing optimal degree(s) of spline or number of segments
I	scalar/vector containing an indicator of whether the predictor is included or not for each dimension of the nominal/ordinal (factor/ordered) predictors
K.mat	vector/matrix of values of K evaluated during search
cv.func	objective function value at optimum
cv.func.vec	vector of objective function values at each degree of spline or number of segments in K.mat

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Craven, P. and G. Wahba (1979), "Smoothing Noisy Data With Spline Functions," Numerische Mathematik, 13, 377-403.

Hurvich, C.M. and J.S. Simonoff and C.L. Tsai (1998), "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion," Journal of the Royal Statistical Society B, 60, 271-293.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

See Also

loess, npregbw,

Examples

```
set.seed(42)
## Simulated data
n <- 1000

x <- runif(n)
z <- round(runif(n,min=-0.5,max=1.5))
z.unique <- uniquecombs(as.matrix(z))
ind <-  attr(z.unique,"index")
ind.vals <-  sort(unique(ind))
dgp <- numeric(length=n)
for(i in 1:nrow(z.unique)) {
  zz <- ind == ind.vals[i]
  dgp[zz] <- z[zz]+cos(2*pi*x[zz])
}

y <- dgp + rnorm(n,sd=.1)

xdata <- data.frame(x,z=factor(z))

## Compute the optimal K and I, determine optimal number of knots, set
## spline degree for x to 3

cv <- frscv(x=xdata,y=y,complexity="knots",degree=c(3))
summary(cv)
```

---

frscvNOMAD *Categorical Factor Regression Spline Cross-Validation*

Description

frscvNOMAD computes NOMAD-based (Nonsmooth Optimization by Mesh Adaptive Direct Search, Abramson, Audet, Couture and Le Digabel (2011)) cross-validation directed search for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous predictors and nominal/ordinal (factor/ordered) predictors.

Usage

```
frscvNOMAD(xz,
           y,
           degree.max = 10,
           segments.max = 10,
           degree.min = 0,
           segments.min = 1,
           cv.df.min = 1,
           complexity = c("degree-knots","degree","knots"),
           knots = c("quantiles","uniform","auto"),
           basis = c("additive","tensor","glp","auto"),
```

```
cv.func = c("cv.ls","cv.gcv","cv.aic"),
degree = degree,
segments = segments,
include = include,
random.seed = 42,
max.bb.eval = 10000,
initial.mesh.size.integer = "1",
min.mesh.size.integer = paste("r",sqrt(.Machine$double.eps),sep=""),
min.poll.size.integer = paste("r",sqrt(.Machine$double.eps),sep=""),
opts=list(),
nmulti = 0,
tau = NULL,
weights = NULL,
singular.ok = FALSE)
```

Arguments

y	continuous univariate vector
xz	continuous and/or nominal/ordinal (factor/ordered) predictors
degree.max	the maximum degree of the B-spline basis for each of the continuous predictors (default degree.max=10)
segments.max	the maximum segments of the B-spline basis for each of the continuous predictors (default segments.max=10)
degree.min	the minimum degree of the B-spline basis for each of the continuous predictors (default degree.min=0)
segments.min	the minimum segments of the B-spline basis for each of the continuous predictors (default segments.min=1)
cv.df.min	the minimum degrees of freedom to allow when conducting cross-validation (default cv.df.min=1)
complexity	a character string (default complexity="degree-knots") indicating whether model 'complexity' is determined by the degree of the spline or by the number of segments ('knots'). This option allows the user to use cross-validation to select either the spline degree (number of knots held fixed) or the number of knots (spline degree held fixed) or both the spline degree and number of knots
knots	a character string (default knots="quantiles") specifying where knots are to be placed. 'quantiles' specifies knots placed at equally spaced quantiles (equal number of observations lie in each segment) and 'uniform' specifies knots placed at equally spaced intervals. If knots="auto", the knot type will be automatically determined by cross-validation
basis	a character string (default basis="additive") indicating whether the additive or tensor product B-spline basis matrix for a multivariate polynomial spline or generalized B-spline polynomial basis should be used. Note this can be automatically determined by cross-validation if cv=TRUE and basis="auto", and is an 'all or none' proposition (i.e. interaction terms for all predictors or for no predictors given the nature of 'tensor products'). Note also that if there is only one predictor this defaults to basis="additive" to avoid unnecessary computation as the spline bases are equivalent in this case

cv.func	a character string (default cv.func="cv.ls") indicating which method to use to select smoothing parameters. cv.gcv specifies generalized cross-validation (Craven and Wahba (1979)), cv.aic specifies expected Kullback-Leibler cross-validation (Hurvich, Simonoff, and Tsai (1998)), and cv.ls specifies least-squares cross-validation
degree	integer/vector specifying the degree of the B-spline basis for each dimension of the continuous x
segments	integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous x (i.e. number of knots minus one)
include	integer/vector for the categorical predictors. If it is not NULL, it will be the initial value for the fitting
random.seed	when it is not missing and not equal to 0, the initial points will be generated using this seed when nmulti > 0
max.bb.eval	argument passed to the NOMAD solver (see snomadr for further details)
initial.mesh.size.integer	argument passed to the NOMAD solver (see snomadr for further details)
min.mesh.size.integer	arguments passed to the NOMAD solver (see snomadr for further details)
min.poll.size.integer	arguments passed to the NOMAD solver (see snomadr for further details)
opts	list of optional arguments to be passed to snomadr
nmulti	integer number of times to restart the process of finding extrema of the cross-validation function from different (random) initial points (default nmulti=0)
tau	if non-null a number in (0,1) denoting the quantile for which a quantile regression spline is to be estimated rather than estimating the conditional mean (default tau=NULL)
weights	an optional vector of weights to be used in the fitting process. Should be 'NULL' or a numeric vector. If non-NULL, weighted least squares is used with weights 'weights' (that is, minimizing 'sum(w*e^2)'); otherwise ordinary least squares is used.
singular.ok	a logical value (default singular.ok=FALSE) that, when FALSE, discards singular bases during cross-validation (a check for ill-conditioned bases is performed).

Details

frscvNOMAD computes NOMAD-based cross-validation for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and nominal/ordinal (factor/ordered) predictors. Numerical search for the optimal degree/segments/I is undertaken using snomadr.

The optimal K/I combination is returned along with other results (see below for return values).

For the continuous predictors the regression spline model employs either the additive or tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (http://www.gnu.org/software/gsl/) and the tensor.prod.model.matrix function.

For the nominal/ordinal (factor/ordered) predictors the regression spline model uses indicator basis functions.

Value

frscvNOMAD returns a crscv object. Furthermore, the function summary supports objects of this type. The returned objects have the following components:

K	scalar/vector containing optimal degree(s) of spline or number of segments
I	scalar/vector containing an indicator of whether the predictor is included or not for each dimension of the nominal/ordinal (factor/ordered) predictors
K.mat	vector/matrix of values of K evaluated during search
degree.max	the maximum degree of the B-spline basis for each of the continuous predictors (default degree.max=10)
segments.max	the maximum segments of the B-spline basis for each of the continuous predictors (default segments.max=10)
degree.min	the minimum degree of the B-spline basis for each of the continuous predictors (default degree.min=0)
segments.min	the minimum segments of the B-spline basis for each of the continuous predictors (default segments.min=1)
cv.func	objective function value at optimum
cv.func.vec	vector of objective function values at each degree of spline or number of segments in K.mat

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca> and Zhenghua Nie <niez@mcmaster.ca>

References

Abramson, M.A. and C. Audet and G. Couture and J.E. Dennis Jr. and S. Le Digabel (2011), "The NOMAD project". Software available at http://www.gerad.ca/nomad.

Craven, P. and G. Wahba (1979), "Smoothing Noisy Data With Spline Functions," Numerische Mathematik, 13, 377-403.

Hurvich, C.M. and J.S. Simonoff and C.L. Tsai (1998), "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion," Journal of the Royal Statistical Society B, 60, 271-293.

Le Digabel, S. (2011), "Algorithm 909: NOMAD: Nonlinear Optimization With the MADS Algorithm". ACM Transactions on Mathematical Software, 37(4):44:1-44:15.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

See Also

loess, npregbw,

Examples

```
set.seed(42)
## Simulated data
n <- 1000

x <- runif(n)
z <- round(runif(n,min=-0.5,max=1.5))
z.unique <- uniquecombs(as.matrix(z))
ind <-  attr(z.unique,"index")
ind.vals <-  sort(unique(ind))
dgp <- numeric(length=n)
for(i in 1:nrow(z.unique)) {
  zz <- ind == ind.vals[i]
  dgp[zz] <- z[zz]+cos(2*pi*x[zz])
}

y <- dgp + rnorm(n,sd=.1)

xdata <- data.frame(x,z=factor(z))

## Compute the optimal K and I, determine optimal number of knots, set
## spline degree for x to 3

cv <- frscvNOMAD(x=xdata,y=y,complexity="knots",degree=c(3),segments=c(5))
summary(cv)
```

---

glp.model.matrix *Utility function for constructing generalized polynomial smooths*

Description

Produce model matrices for a generalized polynomial smooth from the model matrices for the marginal bases of the smooth.

Usage

```
glp.model.matrix(X)
```

Arguments

X	a list of model matrices for the marginal bases of a smooth

Details

This function computes a generalized polynomial where the orders of each term entering the polynomial may vary.

Value

A model matrix for a generalized polynomial smooth.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Hall, P. and J.S. Racine (forthcoming), "Cross-Validated Generalized Local Polynomial Regression," Journal of Econometrics.

Examples

```
X <- list(matrix(1:4,2,2),matrix(5:10,2,3))
glp.model.matrix(X)
```

---

gsl.bs *GSL (GNU Scientific Library) B-spline/B-spline Derivatives*

Description

gsl.bs generates the B-spline basis matrix for a polynomial spline and (optionally) the B-spline basis matrix derivative of a specified order with respect to each predictor

Usage

```
gsl.bs(...)
## Default S3 method:
gsl.bs(x,
       degree = 3,
       nbreak = 2,
       deriv = 0,
       x.min = NULL,
       x.max = NULL,
       intercept = FALSE,
       knots = NULL,
       ...)
```

Arguments

x	the predictor variable. Missing values are not allowed
degree	degree of the piecewise polynomial - default is '3' (cubic spline)
nbreak	number of breaks in each interval - default is '2'
deriv	the order of the derivative to be computed-default if 0
x.min	the lower bound on which to construct the spline - defaults to min(x)

x.max	the upper bound on which to construct the spline - defaults to max(x)
intercept	if 'TRUE', an intercept is included in the basis; default is 'FALSE'
knots	a vector (default knots="NULL") specifying knots for the spline basis (default enables uniform knots, otherwise those provided are used)
...	optional arguments

Details

Typical usages are (see below for a list of options and also the examples at the end of this help file)

```
B <- gsl.bs(x,degree=10)
```

Value

gsl.bs returns a gsl.bs object. A matrix of dimension 'c(length(x), degree+nbreak-1)'.

A primary use is in modelling formulas to directly specify a piecewise polynomial term in a model. See http://www.gnu.org/software/gsl/ for further details.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

See Also

bs

Examples

```
## Plot the spline bases and their first order derivatives
x <- seq(0,1,length=100)
matplot(x,gsl.bs(x,degree=5),type="l")
matplot(x,gsl.bs(x,degree=5,deriv=1),type="l")

## Regression example
n <- 1000
x <- sort(runif(n))
y <- cos(2*pi*x) + rnorm(n,sd=.25)
B <- gsl.bs(x,degree=5,intercept=FALSE)
plot(x,y,cex=.5,col="grey")
lines(x,fitted(lm(y~B)))
```

krscv *Categorical Kernel Regression Spline Cross-Validation*

Description

krscv computes exhaustive cross-validation directed search for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and nominal/ordinal (factor/ordered) predictors.

Usage

```
krscv(xz,
      y,
      degree.max = 10,
      segments.max = 10,
      degree.min = 0,
      segments.min = 1,
      restarts = 0,
      complexity = c("degree-knots","degree","knots"),
      knots = c("quantiles","uniform","auto"),
      basis = c("additive","tensor","glp","auto"),
      cv.func = c("cv.ls","cv.gcv","cv.aic"),
      degree = degree,
      segments = segments,
      tau = NULL,
      weights = NULL,
      singular.ok = FALSE)
```

Arguments

y	continuous univariate vector
xz	continuous and/or nominal/ordinal (factor/ordered) predictors
degree.max	the maximum degree of the B-spline basis for each of the continuous predictors (default degree.max=10)
segments.max	the maximum segments of the B-spline basis for each of the continuous predictors (default segments.max=10)
degree.min	the minimum degree of the B-spline basis for each of the continuous predictors (default degree.min=0)
segments.min	the minimum segments of the B-spline basis for each of the continuous predictors (default segments.min=1)
restarts	number of times to restart optim from different initial random values (default restarts=0) when searching for optimal bandwidths for the categorical predictors for each unique K combination (i.e.\ degree/segments)

`complexity` a character string (default `complexity="degree-knots"`) indicating whether model 'complexity' is determined by the degree of the spline or by the number of segments ('knots'). This option allows the user to use cross-validation to select either the spline degree (number of knots held fixed) or the number of knots (spline degree held fixed) or both the spline degree and number of knots

`knots` a character string (default `knots="quantiles"`) specifying where knots are to be placed. 'quantiles' specifies knots placed at equally spaced quantiles (equal number of observations lie in each segment) and 'uniform' specifies knots placed at equally spaced intervals. If `knots="auto"`, the knot type will be automatically determined by cross-validation

`basis` a character string (default `basis="additive"`) indicating whether the additive or tensor product B-spline basis matrix for a multivariate polynomial spline or generalized B-spline polynomial basis should be used. Note this can be automatically determined by cross-validation if `cv=TRUE` and `basis="auto"`, and is an 'all or none' proposition (i.e. interaction terms for all predictors or for no predictors given the nature of 'tensor products'). Note also that if there is only one predictor this defaults to `basis="additive"` to avoid unnecessary computation as the spline bases are equivalent in this case

`cv.func` a character string (default `cv.func="cv.ls"`) indicating which method to use to select smoothing parameters. `cv.gcv` specifies generalized cross-validation (Craven and Wahba (1979)), `cv.aic` specifies expected Kullback-Leibler cross-validation (Hurvich, Simonoff, and Tsai (1998)), and `cv.ls` specifies least-squares cross-validation

`degree` integer/vector specifying the degree of the B-spline basis for each dimension of the continuous `x`

`segments` integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous `x` (i.e. number of knots minus one)

`tau` if non-null a number in (0,1) denoting the quantile for which a quantile regression spline is to be estimated rather than estimating the conditional mean (default `tau=NULL`)

`weights` an optional vector of weights to be used in the fitting process. Should be 'NULL' or a numeric vector. If non-NULL, weighted least squares is used with weights 'weights' (that is, minimizing 'sum(w*e^2)'); otherwise ordinary least squares is used.

`singular.ok` a logical value (default `singular.ok=FALSE`) that, when `FALSE`, discards singular bases during cross-validation (a check for ill-conditioned bases is performed).

Details

`krscv` computes exhaustive cross-validation for a regression spline estimate of a one (1) dimensional dependent variable on an `r`-dimensional vector of continuous and nominal/ordinal (`factor`/`ordered`) predictors. The optimal `K`/`lambda` combination is returned along with other results (see below for return values). The method uses kernel functions appropriate for categorical (ordinal/nominal) predictors which avoids the loss in efficiency associated with sample-splitting procedures that are typically used when faced with a mix of continuous and nominal/ordinal (`factor`/`ordered`) predictors.

For the continuous predictors the regression spline model employs either the additive or tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (http://www.gnu.org/software/gsl/) and the tensor.prod.model.matrix function.

For the discrete predictors the product kernel function is of the 'Li-Racine' type (see Li and Racine (2007) for details).

For each unique combination of degree and segment, numerical search for the bandwidth vector lambda is undertaken using optim and the box-constrained L-BFGS-B method (see optim for details). The user may restart the optim algorithm as many times as desired via the restarts argument. The approach ascends from K=0 through degree.max/segments.max and for each value of K searches for the optimal bandwidths for this value of K. After the most complex model has been searched then the optimal K/lambda combination is selected. If any element of the optimal K vector coincides with degree.max/segments.max a warning is produced and the user ought to restart their search with a larger value of degree.max/segments.max.

Value

krscv returns a crscv object. Furthermore, the function summary supports objects of this type. The returned objects have the following components:

K	scalar/vector containing optimal degree(s) of spline or number of segments
K.mat	vector/matrix of values of K evaluated during search
restarts	number of restarts during search, if any
lambda	optimal bandwidths for categorical predictors
lambda.mat	vector/matrix of optimal bandwidths for each degree of spline
cv.func	objective function value at optimum
cv.func.vec	vector of objective function values at each degree of spline or number of segments in K.mat

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca>

References

Craven, P. and G. Wahba (1979), "Smoothing Noisy Data With Spline Functions," Numerische Mathematik, 13, 377-403.

Hurvich, C.M. and J.S. Simonoff and C.L. Tsai (1998), "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion," Journal of the Royal Statistical Society B, 60, 271-293.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

See Also

loess, npregbw,

Examples

```
set.seed(42)
## Simulated data
n <- 1000

x <- runif(n)
z <- round(runif(n,min=-0.5,max=1.5))
z.unique <- uniquecombs(as.matrix(z))
ind <-  attr(z.unique,"index")
ind.vals <-  sort(unique(ind))
dgp <- numeric(length=n)
for(i in 1:nrow(z.unique)) {
  zz <- ind == ind.vals[i]
  dgp[zz] <- z[zz]+cos(2*pi*x[zz])
}
y <- dgp + rnorm(n,sd=.1)

xdata <- data.frame(x,z=factor(z))

## Compute the optimal K and lambda, determine optimal number of knots, set
## spline degree for x to 3

cv <- krscv(x=xdata,y=y,complexity="knots",degree=c(3))
summary(cv)
```

---

krscvNOMAD *Categorical Kernel Regression Spline Cross-Validation*

Description

krscvNOMAD computes NOMAD-based (Nonsmooth Optimization by Mesh Adaptive Direct Search, Abramson, Audet, Couture and Le Digabel (2011)) cross-validation directed search for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and nominal/ordinal (factor/ordered) predictors.

Usage

```
krscvNOMAD(xz,
           y,
           degree.max = 10,
           segments.max = 10,
           degree.min = 0,
           segments.min = 1,
           cv.df.min = 1,
           complexity = c("degree-knots","degree","knots"),
```

```
knots = c("quantiles","uniform","auto"),
basis = c("additive","tensor","glp","auto"),
cv.func = c("cv.ls","cv.gcv","cv.aic"),
degree = degree,
segments = segments,
lambda = lambda,
lambda.discrete = FALSE,
lambda.discrete.num = 100,
random.seed = 42,
max.bb.eval = 10000,
initial.mesh.size.real = "r0.1",
initial.mesh.size.integer = "1",
min.mesh.size.real = paste("r",sqrt(.Machine$double.eps),sep=""),
min.mesh.size.integer = paste("r",sqrt(.Machine$double.eps),sep=""),
min.poll.size.real = paste("r",sqrt(.Machine$double.eps),sep=""),
min.poll.size.integer = paste("r",sqrt(.Machine$double.eps),sep=""),
opts=list(),
nmulti = 0,
tau = NULL,
weights = NULL,
singular.ok = FALSE)
```

Arguments

y	continuous univariate vector
xz	continuous and/or nominal/ordinal (factor/ordered) predictors
degree.max	the maximum degree of the B-spline basis for each of the continuous predictors (default degree.max=10)
segments.max	the maximum segments of the B-spline basis for each of the continuous predictors (default segments.max=10)
degree.min	the minimum degree of the B-spline basis for each of the continuous predictors (default degree.min=0)
segments.min	the minimum segments of the B-spline basis for each of the continuous predictors (default segments.min=1)
cv.df.min	the minimum degrees of freedom to allow when conducting cross-validation (default cv.df.min=1)
complexity	a character string (default complexity="degree-knots") indicating whether model 'complexity' is determined by the degree of the spline or by the number of segments ('knots'). This option allows the user to use cross-validation to select either the spline degree (number of knots held fixed) or the number of knots (spline degree held fixed) or both the spline degree and number of knots
knots	a character string (default knots="quantiles") specifying where knots are to be placed. 'quantiles' specifies knots placed at equally spaced quantiles (equal number of observations lie in each segment) and 'uniform' specifies knots placed at equally spaced intervals. If knots="auto", the knot type will be automatically determined by cross-validation

`basis` a character string (default `basis="additive"`) indicating whether the additive or tensor product B-spline basis matrix for a multivariate polynomial spline or generalized B-spline polynomial basis should be used. Note this can be automatically determined by cross-validation if `cv=TRUE` and `basis="auto"`, and is an 'all or none' proposition (i.e. interaction terms for all predictors or for no predictors given the nature of 'tensor products'). Note also that if there is only one predictor this defaults to `basis="additive"` to avoid unnecessary computation as the spline bases are equivalent in this case

`cv.func` a character string (default `cv.func="cv.ls"`) indicating which method to use to select smoothing parameters. `cv.gcv` specifies generalized cross-validation (Craven and Wahba (1979)), `cv.aic` specifies expected Kullback-Leibler cross-validation (Hurvich, Simonoff, and Tsai (1998)), and `cv.ls` specifies least-squares cross-validation

`degree` integer/vector specifying the degree of the B-spline basis for each dimension of the continuous `x`

`segments` integer/vector specifying the number of segments of the B-spline basis for each dimension of the continuous `x` (i.e. number of knots minus one)

`lambda` real/vector for the categorical predictors. If it is not NULL, it will be the starting value(s) for lambda

`lambda.discrete` if `lambda.discrete=TRUE`, the bandwidth will be discretized into `lambda.discrete.num+1` points and `lambda` will be chosen from these points

`lambda.discrete.num` a positive integer indicating the number of discrete values that lambda can assume - this parameter will only be used when `lambda.discrete=TRUE`

`random.seed` when it is not missing and not equal to 0, the initial points will be generated using this seed when `nmulti > 0`

`max.bb.eval` argument passed to the NOMAD solver (see `snomadr` for further details)

`initial.mesh.size.real` argument passed to the NOMAD solver (see `snomadr` for further details)

`initial.mesh.size.integer` argument passed to the NOMAD solver (see `snomadr` for further details)

`min.mesh.size.real` argument passed to the NOMAD solver (see `snomadr` for further details)

`min.mesh.size.integer` arguments passed to the NOMAD solver (see `snomadr` for further details)

`min.poll.size.real` arguments passed to the NOMAD solver (see `snomadr` for further details)

`min.poll.size.integer` arguments passed to the NOMAD solver (see `snomadr` for further details)

`opts` list of optional arguments to be passed to `snomadr`

`nmulti` integer number of times to restart the process of finding extrema of the cross-validation function from different (random) initial points (default `nmulti=0`)

tau	if non-null a number in (0,1) denoting the quantile for which a quantile regression spline is to be estimated rather than estimating the conditional mean (default tau=NULL)
weights	an optional vector of weights to be used in the fitting process. Should be 'NULL' or a numeric vector. If non-NULL, weighted least squares is used with weights 'weights' (that is, minimizing 'sum(w*e^2)'); otherwise ordinary least squares is used.
singular.ok	a logical value (default singular.ok=FALSE) that, when FALSE, discards singular bases during cross-validation (a check for ill-conditioned bases is performed).

Details

krscvNOMAD computes NOMAD-based cross-validation for a regression spline estimate of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and nominal/ordinal (factor/ordered) predictors. Numerical search for the optimal degree/segments/lambda is undertaken using snomadr.

The optimal K/lambda combination is returned along with other results (see below for return values). The method uses kernel functions appropriate for categorical (ordinal/nominal) predictors which avoids the loss in efficiency associated with sample-splitting procedures that are typically used when faced with a mix of continuous and nominal/ordinal (factor/ordered) predictors.

For the continuous predictors the regression spline model employs either the additive or tensor product B-spline basis matrix for a multivariate polynomial spline via the B-spline routines in the GNU Scientific Library (http://www.gnu.org/software/gsl/) and the tensor.prod.model.matrix function.

For the discrete predictors the product kernel function is of the 'Li-Racine' type (see Li and Racine (2007) for details).

Value

krscvNOMAD returns a crscv object. Furthermore, the function summary supports objects of this type. The returned objects have the following components:

K	scalar/vector containing optimal degree(s) of spline or number of segments
K.mat	vector/matrix of values of K evaluated during search
degree.max	the maximum degree of the B-spline basis for each of the continuous predictors (default degree.max=10)
segments.max	the maximum segments of the B-spline basis for each of the continuous predictors (default segments.max=10)
degree.min	the minimum degree of the B-spline basis for each of the continuous predictors (default degree.min=0)
segments.min	the minimum segments of the B-spline basis for each of the continuous predictors (default segments.min=1)
restarts	number of restarts during search, if any
lambda	optimal bandwidths for categorical predictors

lambda.mat	vector/matrix of optimal bandwidths for each degree of spline
cv.func	objective function value at optimum
cv.func.vec	vector of objective function values at each degree of spline or number of segments in K.mat

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca> and Zhenghua Nie <niez@mcmaster.ca>

References

Abramson, M.A. and C. Audet and G. Couture and J.E. Dennis Jr. and S. Le Digabel (2011), "The NOMAD project". Software available at http://www.gerad.ca/nomad.

Craven, P. and G. Wahba (1979), "Smoothing Noisy Data With Spline Functions," Numerische Mathematik, 13, 377-403.

Hurvich, C.M. and J.S. Simonoff and C.L. Tsai (1998), "Smoothing Parameter Selection in Nonparametric Regression Using an Improved Akaike Information Criterion," Journal of the Royal Statistical Society B, 60, 271-293.

Le Digabel, S. (2011), "Algorithm 909: NOMAD: Nonlinear Optimization With The MADS Algorithm". ACM Transactions on Mathematical Software, 37(4):44:1-44:15.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Ma, S. and J.S. Racine and L. Yang (under revision), "Spline Regression in the Presence of Categorical Predictors," Journal of Applied Econometrics.

Ma, S. and J.S. Racine (2013), "Additive Regression Splines with Irrelevant Categorical and Continuous Regressors," Statistica Sinica, Volume 23, 515-541.

See Also

loess, npregbw,

Examples

```
set.seed(42)
## Simulated data
n <- 1000

x <- runif(n)
z <- round(runif(n,min=-0.5,max=1.5))
z.unique <- uniquecombs(as.matrix(z))
ind <-  attr(z.unique,"index")
ind.vals <-  sort(unique(ind))
dgp <- numeric(length=n)
for(i in 1:nrow(z.unique)) {
  zz <- ind == ind.vals[i]
  dgp[zz] <- z[zz]+cos(2*pi*x[zz])
}
y <- dgp + rnorm(n,sd=.1)
```

```
xdata <- data.frame(x,z=factor(z))

## Compute the optimal K and lambda, determine optimal number of knots, set
## spline degree for x to 3

cv <- krscvNOMAD(x=xdata,y=y,complexity="knots",degree=c(3),segments=c(5))
summary(cv)
```

---

npglpreg *Generalized Local Polynomial Regression*

Description

npglpreg computes a generalized local polynomial kernel regression estimate (Hall and Racine (forthcoming)) of a one (1) dimensional dependent variable on an r-dimensional vector of continuous and categorical (factor/ordered) predictors.

Usage

```
npglpreg(...)

## Default S3 method:
npglpreg(tydat = NULL,
         txdat = NULL,
         eydat = NULL,
         exdat = NULL,
         bws = NULL,
         degree = NULL,
         leave.one.out = FALSE,
        ckertype = c("gaussian", "epanechnikov", "uniform", "truncated gaussian"),
         ckerorder = 2,
         ukertype = c("liracine", "aitchisonaitken"),
         okertype = c("liracine", "wangvanryzin"),
         bwtype = c("fixed", "generalized_nn", "adaptive_nn", "auto"),
         gradient.vec = NULL,
         gradient.categorical = FALSE,
         cv.shrink = TRUE,
         cv.maxPenalty = sqrt(.Machine$double.xmax),
         cv.warning = FALSE,
         Bernstein = TRUE,
         mpi = FALSE,
         ...)

## S3 method for class 'formula'
npglpreg(formula,
         data = list(),
         tydat = NULL,
```

```
  txdat = NULL,
  eydat = NULL,
  exdat = NULL,
  bws = NULL,
  degree = NULL,
  leave.one.out = FALSE,
 ckertype = c("gaussian", "epanechnikov","uniform","truncated gaussian"),
  ckerorder = 2,
  ukertype = c("liracine", "aitchisonaitken"),
  okertype = c("liracine", "wangvanryzin"),
  bwtype = c("fixed", "generalized_nn", "adaptive_nn", "auto"),
  cv = c("degree-bandwidth", "bandwidth", "none"),
  cv.func = c("cv.ls", "cv.aic"),
  nmulti = 5,
  random.seed = 42,
  degree.max = 10,
  degree.min = 0,
  bandwidth.max = .Machine$double.xmax,
  bandwidth.min = sqrt(.Machine$double.eps),
  bandwidth.min.numeric = 1.0e-02,
  bandwidth.switch = 1.0e+06,
  bandwidth.scale.categorical = 1.0e+04,
  max.bb.eval = 10000,
  min.epsilon = .Machine$double.eps,
  initial.mesh.size.real = 1,
  initial.mesh.size.integer = 1,
  min.mesh.size.real = sqrt(.Machine$double.eps),
  min.mesh.size.integer = sqrt(.Machine$double.eps),
  min.poll.size.real = sqrt(.Machine$double.eps),
  min.poll.size.integer = sqrt(.Machine$double.eps),
  opts=list(),
  restart.from.min = FALSE,
  gradient.vec = NULL,
  gradient.categorical = FALSE,
  cv.shrink = TRUE,
  cv.maxPenalty = sqrt(.Machine$double.xmax),
  cv.warning = FALSE,
  Bernstein = TRUE,
  mpi = FALSE,
  ...)
```

Arguments

formula	a symbolic description of the model to be fit
data	an optional data frame containing the variables in the model
tydat	a one (1) dimensional numeric or integer vector of dependent data, each element i corresponding to each observation (row) i of txdat. Defaults to the training data used to compute the bandwidth object

txdat	a p-variate data frame of explanatory data (training data) used to calculate the regression estimators. Defaults to the training data used to compute the bandwidth object
eydat	a one (1) dimensional numeric or integer vector of the true values of the dependent variable. Optional, and used only to calculate the true errors
exdat	a p-variate data frame of points on which the regression will be estimated (evaluation data). By default, evaluation takes place on the data provided by txdat
bws	a vector of bandwidths, with each element i corresponding to the bandwidth for column i in txdat
degree	integer/vector specifying the polynomial degree of the for each dimension of the continuous x in txdat
leave.one.out	a logical value to specify whether or not to compute the leave one out sums. Will not work if exdat is specified. Defaults to FALSE
ckertype	character string used to specify the continuous kernel type. Can be set as gaussian, epanechnikov, or uniform. Defaults to gaussian.
ckerorder	numeric value specifying kernel order (one of (2,4,6,8)). Kernel order specified along with a uniform continuous kernel type will be ignored. Defaults to 2.
ukertype	character string used to specify the unordered categorical kernel type. Can be set as aitchisonaitken or liracine. Defaults to liracine
okertype	character string used to specify the ordered categorical kernel type. Can be set as wangvanryzin or liracine. Defaults to liracine
bwtype	character string used for the continuous variable bandwidth type, specifying the type of bandwidth to compute and return in the bandwidth object. If bwtype="auto", the bandwidth type type will be automatically determined by cross-validation. Defaults to fixed. Option summary: fixed: compute fixed bandwidths generalized_nn: compute generalized nearest neighbor bandwidths adaptive_nn: compute adaptive nearest neighbor bandwidths
cv	a character string (default cv="nomad") indicating whether to use nonsmooth mesh adaptive direct search, or no search (i.e. use supplied values for degree and bws)
cv.func	a character string (default cv.func="cv.ls") indicating which method to use to select smoothing parameters. cv.aic specifies expected Kullback-Leibler cross-validation (Hurvich, Simonoff, and Tsai (1998)), and cv.ls specifies least-squares cross-validation
max.bb.eval	argument passed to the NOMAD solver (see snomadr for further details)
min.epsilon	argument passed to the NOMAD solver (see snomadr for further details)
initial.mesh.size.real	argument passed to the NOMAD solver (see snomadr for further details)
initial.mesh.size.integer	argument passed to the NOMAD solver (see snomadr for further details)
min.mesh.size.real	argument passed to the NOMAD solver (see snomadr for further details)

`min.mesh.size.integer`
: arguments passed to the NOMAD solver (see `snomadr` for further details)

`min.poll.size.real`
: arguments passed to the NOMAD solver (see `snomadr` for further details)

`min.poll.size.integer`
: arguments passed to the NOMAD solver (see `snomadr` for further details)

`opts`
: list of optional arguments passed to the NOMAD solver (see `snomadr` for further details)

`nmulti`
: integer number of times to restart the process of finding extrema of the cross-validation function from different (random) initial points (default `nmulti=5`)

`random.seed`
: when it is not missing and not equal to 0, the initial points will be generated using this seed when using `snomadr`

`degree.max`
: the maximum degree of the polynomial for each of the continuous predictors (default `degree.max=10`)

`degree.min`
: the minimum degree of the polynomial for each of the continuous predictors (default `degree.min=0`)

`bandwidth.max`
: the maximum bandwidth scale (i.e. number of scaled standard deviations) for each of the continuous predictors (default `bandwidth.max=.Machine$double.xmax`)

`bandwidth.min`
: the minimum bandwidth scale for each of the categorical predictors (default `sqrt(.Machine$double.eps)`)

`bandwidth.min.numeric`
: the minimum bandwidth scale (i.e. number of scaled standard deviations) for each of the continuous predictors (default `bandwidth.min=1.0e-02`)

`bandwidth.switch`
: the minimum bandwidth scale (i.e. number of scaled standard deviations) for each of the continuous predictors (default `bandwidth.switch=1.0e+06`) before the local polynomial is treated as global during cross-validation at which point a global categorical kernel weighted least squares fit is used for computational efficiency

`bandwidth.scale.categorical`
: the upper end for the rescaled bandwidths for the categorical predictors (default `bandwidth.scale.categorical=1.0e+04`) - the aim is to 'even up' the scale of the search parameters as much as possible, so when very large scale factors are selected for the continuous predictors, a larger value may improve search

`restart.from.min`
: a logical value indicating to recommence `snomadr` with the optimal values found from its first invocation (typically quick but sometimes recommended in the field of optimization)

`gradient.vec`
: a vector corresponding to the order of the partial (or cross-partial) and which variable the partial (or cross-partial) derivative(s) are required

`gradient.categorical`
: a logical value indicating whether discrete gradients (i.e. differences in the response from the base value for each categorical predictor) are to be computed

cv.shrink	a logical value indicating whether to use ridging (Seifert and Gasser (2000)) for ill-conditioned inversion during cross-validation (default cv.shrink=TRUE) or to instead test for ill-conditioned matrices and penalize heavily when this is the case (much stronger condition imposed on cross-validation)
cv.maxPenalty	a penalty applied during cross-validation when a delete-one estimate is not finite or the polynomial basis is not of full column rank
cv.warning	a logical value indicating whether to issue an immediate warning message when ill conditioned bases are encountered during cross-validation (default cv.warning=FALSE)
Bernstein	a logical value indicating whether to use raw polynomials or Bernstein polynomials (default) (note that a Bernstein polynomial is also know as a Bezier curve which is also a B-spline with no interior knots)
mpi	a logical value (default mpi=FALSE) that, when mpi=TRUE, can call the npRmpi rather than the np package (note - code needs to mirror examples in the demo directory of the npRmpi package, you need to broadcast loading of the crs package, and need .Rprofile in your current directory)
...	additional arguments supplied to specify the regression type, bandwidth type, kernel types, training data, and so on, detailed below

Details

Typical usages are (see below for a list of options and also the examples at the end of this help file)

```
## Conduct generalized local polynomial estimation

model <- npglpreg(y~x1+x2)

## Conduct degree 0 local polynomial estimation
## (i.e. Nadaraya-Watson)

model <- npglpreg(y~x1+x2,cv="bandwidth",degree=c(0,0))

## Conduct degree 1 local polynomial estimation (i.e. local linear)

model <- npglpreg(y~x1+x2,cv="bandwidth",degree=c(1,1))

## Conduct degree 2 local polynomial estimation (i.e. local
## quadratic)

model <- npglpreg(y~x1+x2,cv="bandwidth",degree=c(2,2))

## Plot the mean and bootstrap confidence intervals

plot(model,ci=TRUE)

## Plot the first partial derivatives and bootstrap confidence
## intervals
```

```
plot(model,deriv=1,ci=TRUE)

## Plot the first second partial derivatives and bootstrap
## confidence intervals

plot(model,deriv=2,ci=TRUE)
```

This function is in beta status until further notice (eventually it will be rolled into the np/npRmpi packages after the final status of snomadr/NOMAD gets sorted out).

Optimizing the cross-validation function jointly for bandwidths (vectors of continuous parameters) and polynomial degrees (vectors of integer parameters) constitutes a mixed-integer optimization problem. These problems are not only 'hard' from the numerical optimization perspective, but are also computationally intensive (contrast this to where we conduct, say, local linear regression which sets the degree of the polynomial vector to a global value `degree=1` hence we only need to optimize with respect to the continuous bandwidths). Because of this we must be mindful of the presence of local optima (the objective function is non-convex and non-differentiable). Restarting search from different initial starting points is recommended (see `nmulti`) and by default this is done more than once. We encourage users to adopt 'multistarting' and to investigate the impact of changing default search parameters such as `initial.mesh.size.real`, `initial.mesh.size.integer`, `min.mesh.size.real`, `min.mesh.size.integer`,`min.poll.size.real`, and `min.poll.size.integer`. The default values were chosen based on extensive simulation experiments and were chosen so as to yield robust performance while being mindful of excessive computation - of course, no one setting can be globally optimal.

Value

`npglpreg` returns a `npglpreg` object. The generic functions `fitted` and `residuals` extract (or generate) estimated values and residuals. Furthermore, the functions `summary`, `predict`, and `plot` (options `deriv=0`, `ci=FALSE` [`ci=TRUE` produces pointwise bootstrap error bounds], `persp.rgl=FALSE`, `plot.behavior=c("plot","plot-data","data")`, `plot.errors.boot.num=99`, `plot.errors.type=c("quantiles","s` ["quantiles" produces percentiles determined by `plot.errors.quantiles` below, "standard" produces error bounds given by +/- 1.96 bootstrap standard deviations], `plot.errors.quantiles=c(.025,.975)`, `xtrim=0.0`, `xq=0.5`) support objects of this type. The returned object has the following components:

`fitted.values`	estimates of the regression function (conditional mean) at the sample points or evaluation points
`residuals`	residuals computed at the sample points or evaluation points
`degree`	integer/vector specifying the degree of the polynomial for each dimension of the continuous `x`
`gradient`	the estimated gradient (vector) corresponding to the vector `gradient.vec`
`gradient.categorical.mat`	the estimated gradient (matrix) for the categorical predictors
`gradient.vec`	the supplied `gradient.vec`
`bws`	vector of bandwidths

bwtype	the supplied bwtype
call	a symbolic description of the model
r.squared	coefficient of determination (Doksum and Samarov (1995))

Note

Note that the use of raw polynomials (Bernstein=FALSE) for approximation is appealing as they can be computed and differentiated easily, however, they can be unstable (their inversion can be ill conditioned) which can cause problems in some instances as the order of the polynomial increases. This can hamper search when excessive reliance on ridging to overcome ill conditioned inversion becomes computationally burdensome.

npglpreg tries to detect whether this is an issue or not when Bernstein=FALSE for each numeric predictor and will adjust the search range for snomadr and the degree fed to npglpreg if appropriate.

However, if you suspect that this might be an issue for your specific problem and you are using raw polynomials (Bernstein=FALSE), you are encouraged to investigate this by limiting degree.max to value less than the default value (say 3). Alternatively, you might consider re-scaling your numeric predictors to lie in $[0, 1]$ using scale.

For a given predictor x you can readily determine if this is an issue by considering the following: Suppose x is given by

```
x <- runif(100,10000,11000)
y <- x + rnorm(100,sd=1000)
```

so that a polynomial of order, say, 5 would be ill conditioned. This would be apparent if you considered

```
X <- poly(x,degree=5,raw=TRUE)
solve(t(X)%*%X)
```

which will throw an error when the polynomial is ill conditioned, or

```
X <- poly(x,degree=5,raw=TRUE)
lm(y~X)
```

which will return NA for one or more coefficients when this is an issue.

In such cases you might consider transforming your numeric predictors along the lines of the following:

```
x <- as.numeric(scale(x))
X <- poly(x,degree=5,raw=TRUE)
solve(t(X)%*%X)
lm(y~X)
```

Note that now your least squares coefficients (i.e. first derivative of y with respect to x) represent the effect of a one standard deviation change in x and not a one unit change.

Alternatively, you can use Bernstein polynomials by not setting Bernstein=FALSE.

Author(s)

Jeffrey S. Racine <racinej@mcmaster.ca> and Zhenghua Nie <niez@mcmaster.ca>

References

Doksum, K. and A. Samarov (1995), "Nonparametric Estimation of Global Functionals and a Measure of the Explanatory Power of Covariates in Regression," The Annals of Statistics, 23 1443-1473.

Hall, P. and J.S. Racine (forthcoming), "Cross-Validated Generalized Local Polynomial Regression," Journal of Econometrics.

Li, Q. and J.S. Racine (2007), *Nonparametric Econometrics: Theory and Practice,* Princeton University Press.

Seifert, B. and T. Gasser (2000), "Data Adaptive Ridging in Local Polynomial Regression," Journal of Computational and Graphical Statistics, 9(2), 338-360.

See Also

npreg

Examples

```
## Not run:
set.seed(42)
n <- 100
x1 <- runif(n,-2,2)
x2 <- runif(n,-2,2)
y <- x1^3 + rnorm(n,sd=1)

## Ideally the method should choose large bandwidths for x1 and x2 and a
## generalized polynomial that is a cubic for x1 and degree 0 for x2.

model <- npglpreg(y~x1+x2,nmulti=1)
summary(model)

## Plot the partial means and percentile confidence intervals
plot(model,ci=T)
## Extract the data from the plot object and plot it separately
myplot.dat <- plot(model,plot.behavior="data",ci=T)
matplot(myplot.dat[[1]][,1],myplot.dat[[1]][,-1],type="l")
matplot(myplot.dat[[2]][,1],myplot.dat[[2]][,-1],type="l")

## End(Not run)
```

---

snomadr *R interface to NOMAD*

Description

snomadr is an R interface to NOMAD (Nonsmooth Optimization by Mesh Adaptive Direct Search, Abramson, Audet, Couture and Le Digabel (2011)), an open source software C++ implementation of the Mesh Adaptive Direct Search (MADS, Le Digabel (2011)) algorithm designed for constrained optimization of blackbox functions.

NOMAD is designed to find (local) solutions of mathematical optimization problems of the form

```
   min     f(x)
x in R^n

s.t.       g(x) <= 0
           x_L <=  x   <= x_U
```

where $f(x)$: $R^n \rightarrow R^k$ is the objective function, and $g(x)$: $R^n \rightarrow R^m$ are the constraint functions. The vectors x_L and x_U are the bounds on the variables x. The functions $f(x)$ and $g(x)$ can be nonlinear and nonconvex. The variables can be integer, continuous real number, binary, and categorical.

Kindly see http://www.gerad.ca/nomad/Project/Home.html and the references below for details.

Usage

```
snomadr(eval.f,
        n,
        bbin = NULL,
        bbout = NULL,
        x0 = NULL,
        lb = NULL,
        ub = NULL,
        nmulti = 0,
        random.seed = 0,
        opts = list(),
        print.output = TRUE,
        information = list(),
        snomadr.environment = new.env(),
        ... )
```

Arguments

eval.f	function that returns the value of the objective function
n	the number of variables
bbin	types of variables. Variable types are 0 (CONTINUOUS), 1 (INTEGER), 2 (CATEGORICAL), 3 (BINARY)
bbout	types of output of eval.f. See the NOMAD User Guide http://www.gerad.ca/NOMAD/Downloads/user_guide.pdf
x0	vector with starting values for the optimization. If it is provided and nmulti is bigger than 1, x0 will be the first initial point for multiple initial points

`lb` vector with lower bounds of the controls (use -1.0e19 for controls without lower bound)

`ub` vector with upper bounds of the controls (use 1.0e19 for controls without upper bound)

`nmulti` when it is missing, or it is equal to 0 and `x0` is provided, `snomadRSolve` will be called to solve the problem. Otherwise, `smultinomadRSolve` will be called

`random.seed` when it is not missing and not equal to 0, the initial points will be generated using this seed when `nmulti > 0`

`opts` list of options for NOMAD, see the NOMAD user guide http://www.gerad.ca/NOMAD/Downloads/user_guide.pdf. Options can also be set by nomad.opt which should be in the folder where R (`snomadr`) is executed. Options that affect the solution and their defaults and some potential values are

`"MAX_BB_EVAL"=10000`

`"INITIAL_MESH_SIZE"=1`

`"MIN_MESH_SIZE"="r1.0e-10"`

`"MIN_POLL_SIZE"="r1.0e-10"`

Note that the `"r..."` denotes relative measurement (relative to `lb` and `ub`)

Note that decreasing the maximum number of black box evaluations (`"MAX_BB_EVAL"`) will terminate search sooner and may result in a less accurate solution. For complicated problems you may want to increase this value. When experimenting it is desirable to set `"DISPLAY_DEGREE"=1` (or a larger integer) to get some sense for how the algorithm is progressing

`print.output` when FALSE, no output from `snomadr` is displayed on the screen. If the NOMAD option `"DISPLAY_DEGREE"=0`, is set, there will also be no output from NOMAD. Higher integer values for `"DISPLAY_DEGREE"=` provide successively more detail

`information` is a list. `snomadr` will call `snomadRInfo` to return the information about NOMAD according to the values of `"info"`, `"version"` and `"help"`.

`"info"="-i"`: display the usage and copyright of NOMAD

`"version"="-v"`: display the version of NOMAD you are using

`"help"="-h"`: display all options

You also can display a specific option, for example, `"help"="-h x0"`, this will tell you how to set `x0`

`snomadr.environment`
environment that is used to evaluate the functions. Use this to pass additional data or parameters to a function

`...` arguments that will be passed to the user-defined objective and constraints functions. See the examples below

Details

`snomadr` is used in the **crs** package to numerically minimize an objective function with respect to the spline degree, number of knots, and optionally the kernel bandwidths when using `crs` with the option `cv="nomad"` (default). This is a constrained mixed integer combinatoric problem and

is known to be computationally 'hard'. See frscvNOMAD and krscvNOMAD for the functions called when cv="nomad" while using crs.

However, the user should note that for simple problems involving one predictor exhaustive search may be faster and potentially more accurate, so please bear in mind that cv="exhaustive" can be useful when using crs.

Naturally, exhaustive search is also useful for verifying solutions returned by snomadr. See frscv and krscv for the functions called when cv="exhaustive" while using crs.

Value

The return value contains a list with the inputs, and additional elements

call	the call that was made to solve
status	integer value with the status of the optimization
message	more informative message with the status of the optimization
iterations	number of iterations that were executed, if multiple initial points are set, this number will be the sum for each initial point.
objective	value if the objective function in the solution
solution	optimal value of the controls

Author(s)

Zhenghua Nie <niez@mcmaster.ca>

References

Abramson, M.A. and C. Audet and G. Couture and J.E. Dennis Jr. and S. Le Digabel (2011), "The NOMAD project". Software available at http://www.gerad.ca/nomad.

Le Digabel, S. (2011), "Algorithm 909: NOMAD: Nonlinear Optimization With The MADS Algorithm". ACM Transactions on Mathematical Software, 37(4):44:1-44:15.

See Also

optim, nlm, nlminb

Examples

```
## Not run:
## List all options
snomadr(information=list("help"="-h"))

## Print given option,  for example,  MESH_SIZE
snomadr(information=list("help"="-h MESH_SIZE"))

## Print the version of NOMAD
snomadr(information=list("version"="-v"))

## Print usage and copyright
```

```
snomadr(information=list("info"="-i"))

## This is the example found in
## NOMAD/examples/basic/library/single_obj/basic_lib.cpp

eval.f <- function ( x ) {

    f <- c(Inf, Inf, Inf);
    n <- length (x);

    if ( n == 5 && ( is.double(x) || is.integer(x) ) ) {
        f[1] <- x[5];
        f[2] <- sum ( (x-1)^2 ) - 25;
        f[3] <- 25 - sum ( (x+1)^2 );
    }

    return ( as.double(f) );
}

## Initial values
x0 <- rep(0.0, 5 )

bbin <-c(1, 1, 1, 1, 1)
## Bounds
lb <- rep(-6.0,5 )
ub <- c(5.0, 6.0, 7.0, 1000000, 100000)

bbout <-c(0, 2, 1)
## Options
opts <-list("MAX_BB_EVAL"=500,
            "MIN_MESH_SIZE"=0.001,
            "INITIAL_MESH_SIZE"=0.1,
            "MIN_POLL_SIZE"=0.0001)

snomadr(eval.f=eval.f,n=5,  x0=x0, bbin=bbin, bbout=bbout, lb=lb, ub=ub, opts=opts)


## How to transfer other parameters into eval.f
##
## First example: supply additional arguments in user-defined functions
##

## objective function and gradient in terms of parameters
eval.f.ex1 <- function(x, params) {
    return( params[1]*x^2 + params[2]*x + params[3] )
}

## Define parameters that we want to use
params <- c(1,2,3)

## Define initial value of the optimization problem
x0 <- 0
```

```
## solve using snomadr
snomadr( n          =1,
        x0          = x0,
        eval.f      = eval.f.ex1,
        params      = params )


##
## Second example: define an environment that contains extra parameters
##

## Objective function and gradient in terms of parameters
## without supplying params as an argument
eval.f.ex2 <- function(x) {
    return( params[1]*x^2 + params[2]*x + params[3] )
}

## Define initial value of the optimization problem
x0 <- 0

## Define a new environment that contains params
auxdata        <- new.env()
auxdata$params <- c(1,2,3)

## pass The environment that should be used to evaluate functions to snomadr
snomadr(n                 =1,
        x0                = x0,
        eval.f            = eval.f.ex2,
        snomadr.environment = auxdata )

## Solve using algebra
cat( paste( "Minimizing f(x) = ax^2 + bx + c\n" ) )
cat( paste( "Optimal value of control is -b/(2a) = ", -params[2]/(2*params[1]), "\n" ) )
cat( paste( "With value of the objective function f(-b/(2a)) = ",
          eval.f.ex1( -params[2]/(2*params[1]), params ), "\n" ) )

## The following example is NOMAD/examples/advanced/multi_start/multi.cpp
## This will call smultinomadRSolve to resolve the problem.
eval.f.ex1 <- function(x, params) {
    M<-as.numeric(params$M)
    NBC<-as.numeric(params$NBC)

    f<-rep(0, M+1)
    x<-as.numeric(x)

    x1 <- rep(0.0, NBC)
    y1 <- rep(0.0, NBC)

    x1[1]<-x[1]
    x1[2]<-x[2]
    y1[3]<-x[3]
    x1[4]<-x[4]
    y1[4]<-x[5]
```

```
    epi <- 6
    for(i in 5:NBC){
        x1[i]<-x[epi]
        epi <- epi+1
        y1[i]<-x[epi]
        epi<-epi+1
    }
    constraint <- 0.0
    ic <- 1
    f[ic]<-constraint
    ic <- ic+1

    constraint <- as.numeric(1.0)
    distmax <- as.numeric(0.0)
    avg_dist <- as.numeric(0.0)
    dist1<-as.numeric(0.0)

    for(i in 1:(NBC-1)){
        for (j in (i+1):NBC){
            dist1 <- as.numeric((x1[i]-x1[j])*(x1[i]-x1[j])+(y1[i]-y1[j])*(y1[i]-y1[j]))

            if((dist1 > distmax)) {distmax <- as.numeric(dist1)}
            if((dist1[1]) < 1) {constraint <- constraint*sqrt(dist1)}
            else if((dist1) > 14) {avg_dist <- avg_dist+sqrt(dist1)}
        }
    }

    if(constraint < 0.9999) constraint <- 1001.0-constraint
    else constraint = sqrt(distmax)+avg_dist/(10.0*NBC)

    f[2] <- 0.0
    f[M+1] <- constraint


    return(as.numeric(f) )
}

## Define parameters that we want to use
params<-list()
NBC <- 5
M <- 2
n<-2*NBC-3

params$NBC<-NBC
params$M<-M
x0<-rep(0.1, n)
lb<-rep(0, n)
ub<-rep(4.5, n)

eval.f.ex1(x0, params)

bbout<-c(2, 2, 0)
```

```
nmulti=5
bbin<-rep(0, n)
## Define initial value of the optimization problem

## Solve using snomadRSolve
snomadr(n            = as.integer(n),
        x0           = x0,
        eval.f       = eval.f.ex1,
        bbin         = bbin,
        bbout        = bbout,
        lb           = lb,
        ub           = ub,
        params       = params )

## Solve using smultinomadRSolve, if x0 is provided,  x0 will
## be the first initial point,  otherwise,  the program will
## check best_x.txt,  if it exists,  it will be read in as
## the first initial point. Other initial points will be
## generated by uniform distribution.
## nmulti represents the number of mads will run.
##
snomadr(n            = as.integer(n),
        eval.f       = eval.f.ex1,
        bbin         = bbin,
        bbout        = bbout,
        lb           = lb,
        ub           = ub,
        nmulti = as.integer(nmulti),
        print.output = TRUE,
        params       = params )

## End(Not run)
```

`tensor.prod.model.matrix` *Utility functions for constructing tensor product smooths*

Description

Produce model matrices or penalty matrices for a tensor product smooth from the model matrices or penalty matrices for the marginal bases of the smooth.

Usage

```
tensor.prod.model.matrix(X)
```

Arguments

`X`	a list of model matrices for the marginal bases of a smooth

Details

If `X[[1]]`, `X[[2]]` ... `X[[m]]` are the model matrices of the marginal bases of a tensor product smooth then the ith row of the model matrix for the whole tensor product smooth is given by `X[[1]][i,]%x%X[[2]][i,]%x% ... X[[m]][i,]`, where `%x%` is the Kronecker product. Of course the routine operates column-wise, not row-wise!

Value

Either a single model matrix for a tensor product smooth, or a list of penalty terms for a tensor product smooth.

Author(s)

Simon N. Wood `<simon.wood@r-project.org>`

References

Wood, S.N. (2006) "Low Rank Scale Invariant Tensor Product Smooths for Generalized Additive Mixed Models". Biometrics 62(4):1025-1036

See Also

`te`, `smooth.construct.tensor.smooth.spec`

Examples

```
X <- list(matrix(1:4,2,2),matrix(5:10,2,3))
tensor.prod.model.matrix(X)
```

---

uniquecombs *Find the unique rows in a matrix*

Description

This routine returns a matrix containing all the unique rows of the matrix supplied as its argument. That is, all the duplicate rows are stripped out. Note that the ordering of the rows on exit is not the same as on entry. It also returns an index attribute for relating the result back to the original matrix.

Usage

```
uniquecombs(x)
```

Arguments

x	is an R matrix (numeric)

Details

Models with more parameters than unique combinations of covariates are not identifiable. This routine provides a means of evaluating the number of unique combinations of covariates in a model. The routine calls compiled C code.

Value

A matrix consisting of the unique rows of x (in arbitrary order).

The matrix has an "index" attribute. index[i] gives the row of the returned matrix that contains row i of the original matrix.

Author(s)

Simon N. Wood <simon.wood@r-project.org>

See Also

unique can do the same thing, including for non-numeric matrices, but more slowly and without returning the index.

Examples

```
X<-matrix(c(1,2,3,1,2,3,4,5,6,1,3,2,4,5,6,1,1,1),6,3,byrow=TRUE)
print(X)
Xu <- uniquecombs(X);Xu
ind <- attr(Xu,"index")
## find the value for row 3 of the original from Xu
Xu[ind[3],];X[3,]
```

---

wage1 *Cross-Sectional Data on Wages*

Description

Cross-section wage data consisting of a random sample taken from the U.S. Current Population Survey for the year 1976. There are 526 observations in total.

Usage

```
data("wage1")
```

Format

A data frame with 24 columns, and 526 rows.

wage column 1, of type `numeric`, average hourly earnings

educ column 2, of type `numeric`, years of education

exper column 3, of type `numeric`, years potential experience

tenure column 4, of type `numeric`, years with current employer

nonwhite column 5, of type `factor`, ="Nonwhite" if nonwhite, "White" otherwise

female column 6, of type `factor`, ="Female" if female, "Male" otherwise

married column 7, of type `factor`, ="Married" if Married, "Nonmarried" otherwise

numdep column 8, of type `numeric`, number of dependents

smsa column 9, of type `numeric`, =1 if live in SMSA

northcen column 10, of type `numeric`, =1 if live in north central U.S

south column 11, of type `numeric`, =1 if live in southern region

west column 12, of type `numeric`, =1 if live in western region

construc column 13, of type `numeric`, =1 if work in construc. indus.

ndurman column 14, of type `numeric`, =1 if in nondur. manuf. indus.

trcommpu column 15, of type `numeric`, =1 if in trans, commun, pub ut

trade column 16, of type `numeric`, =1 if in wholesale or retail

services column 17, of type `numeric`, =1 if in services indus.

profserv column 18, of type `numeric`, =1 if in prof. serv. indus.

profocc column 19, of type `numeric`, =1 if in profess. occupation

clerocc column 20, of type `numeric`, =1 if in clerical occupation

servocc column 21, of type `numeric`, =1 if in service occupation

lwage column 22, of type `numeric`, log(wage)

expersq column 23, of type `numeric`, exper^2

tenursq column 24, of type `numeric`, tenure^2

Source

Jeffrey M. Wooldridge

References

Wooldridge, J.M. (2000), *Introductory Econometrics: A Modern Approach*, South-Western College Publishing.

Examples

```
## Not run:
data(wage1)

## Cross-validated model selection for spline degree and bandwidths Note
## - we override the default nmulti here to get a quick illustration
## (we don't advice doing this, in fact advise using more restarts in
## serious applications)

model <- crs(lwage~married+
             female+
             nonwhite+
             educ+
             exper+
             tenure,
             basis="additive",
             complexity="degree",
             data=wage1,
             segments=c(1,1,1),
             nmulti=1)

summary(model)

## Residual plots
plot(model)
## Partial mean plots (control for non axis predictors)
plot(model,mean=TRUE)
## Partial first derivative plots (control for non axis predictors)
plot(model,deriv=1)
## Partial second derivative plots (control for non axis predictors)
plot(model,deriv=2)
## Compare with local linear kernel regression
require(np)
model <- npreg(lwage~married+
               female+
               nonwhite+
               educ+
               exper+
               tenure,
               regtype="ll",
               bwmethod="cv.aic",
               data=wage1)

summary(model)

## Partial mean plots (control for non axis predictors)
plot(model,common.scale=FALSE)
## Partial first derivative plots (control for non axis predictors)
plot(model,gradients=TRUE,common.scale=FALSE)
detach("package:np")

## End(Not run)
```

Index