Package ‘cgam’

May 8, 2026

Type Package

Title Constrained Generalized Additive Model

Version 1.32

Date 2026-03-05

Description A constrained generalized additive model is fitted by the cgam routine. Given a set of predictors, each of which may have a shape or order restrictions, the maximum likelihood estimator for the constrained generalized additive model is found using an iteratively reweighted cone projection algorithm. The ShapeSelect routine chooses a subset of predictor variables and describes the component relationships with the response. For each predictor, the user needs only specify a set of possible shape or order restrictions. A model selection method chooses the shapes and orderings of the relationships as well as the variables. The cone information criterion (CIC) is used to select the best combination of variables and shapes. A genetic algorithm may be used when the set of possible models is large. In addition, the cgam routine implements a two-dimensional isotonic regression using warped-plane splines without additivity assumptions. It can also fit a convex or concave regression surface with triangle splines without additivity assumptions. See Liao X, Meyer MC (2019)<doi:10.18637/jss.v089.i05> for more details.

License GPL (>= 2)

Depends coneproj (>= 1.20), R(>= 4.3.0)

Imports splines2, svDialogs, statmod, lme4, Matrix, ggplot2, dplyr, parallel, zeallot, rlang, quadprog, MASS, stats

RoxygenNote 7.3.2

NeedsCompilation no

Suggests graphics, grDevices, utils, roxygen2, patchwork, glmnet

ByteCompile true

Author Mary Meyer [aut],
Xiyue Liao [aut, cre] (ORCID: <`https://orcid.org/0000-0002-4508-9219`>)

Maintainer Xiyue Liao <`xliao@sdsu.edu`>

Repository CRAN

Date/Publication 2026-03-05 06:10:50 UTC

Contents

`cgam` *Constrained Generalized Additive Model Fitting*

Description

The partial linear generalized additive model is fitted using the method of maximum likelihood, where shape or order restrictions can be imposed on the non-parametrically modelled predictors with optional smoothing, and no restrictions are imposed on the optional parametrically modelled covariate.

Usage

```
cgam(formula, cic = FALSE, nsim = 100, family = gaussian, cpar = 1.5,
data = NULL, weights = NULL, sc_x = FALSE, sc_y = FALSE, pnt = TRUE,
pen = 0, var.est = NULL, gcv = FALSE, pvf = TRUE)
```

Arguments

`formula` A formula object which gives a symbolic description of the model to be fitted. It has the form "response ~ predictor". The response is a vector of length n. The specification of the model can be one of the three exponential families: gaussian, binomial and poisson. The systematic component η is $E(y)$, the log odds of $y = 1$, and the logarithm of $E(y)$ respectively. A predictor can be a non-parametrically modelled variable with or without a shape or order restriction, or a parametrically modelled unconstrained covariate. In terms of a non-parametrically modelled predictor, the user is supposed to indicate the relationship between the systematic component η and a predictor x in the following way:

Assume that η is the systematic component and x is a predictor:

incr(x): η is increasing in x. See `incr` for more details.

s.incr(x): η is smoothly increasing in x. See `s.incr` for more details.

decr(x): η is decreasing in x. See `decr` for more details.

s.decr(x): η is smoothly decreasing in x. See `s.decr` for more details.

conc(x): η is concave in x. See `conc` for more details.

s.conc(x): η is smoothly concave in x. See `s.conc` for more details.

conv(x): η is convex in x. See `conv` for more details.

s.conv(x): η is smoothly convex in x. See `s.conv` for more details.

incr.conc(x): η is increasing and concave in x. See `incr.conc` for more details.

s.incr.conc(x): η is smoothly increasing and concave in x. See `s.incr.conc` for more details.

decr.conc(x): η is decreasing and concave in x. See `decr.conc` for more details.

s.decr.conc(x): η is smoothly decreasing and concave in x. See `s.decr.conc` for more details.

incr.conv(x): η is increasing and convex in x. See `incr.conv` for more details.

s.incr.conv(x): η is smoothly increasing and convex in x. See `s.incr.conv` for more details.

decr.conv(x): η is decreasing and convex in x. See `decr.conv` for more details.

s.decr.conv(x): η is smoothly decreasing and convex in x. See `s.decr.conv` for more details.

s(x): η is smooth in x. See `s` for more details.

tree(x): η has a tree-ordering in x. See `tree` for more details.

umbrella(x): η has an umbrella-ordering in x. See `umbrella` for more details.

`cic`	Logical flag indicating if or not simulations are used to get the cic value. The default is cic = FALSE
`nsim`	The number of simulations used to get the cic parameter. The default is nsim = 100.
`family`	A parameter indicating the error distribution and link function to be used in the model. It can be a character string naming a family function or the result of a call to a family function. This is borrowed from the glm routine in the stats package. There are four families used in csvy: Gaussian, binomial, poisson, and Gamma. Note that if family = Ord is specified, a proportional odds regression model with shape constraints is fitted. This is under development.
`cpar`	A multiplier to estimate the model variance, which is defined as $\sigma^2 = SSR/(n-cpar*edf)$. SSR is the sum of squared residuals for the full model and edf is the effective degrees of freedom. The default is cpar = 1.2. The user-defined value must be between 1 and 2. See Meyer, M. C. and M. Woodroofe (2000) for more details.
`data`	An optional data frame, list or environment containing the variables in the model. The default is data = NULL.
`weights`	An optional non-negative vector of "replicate weights" which has the same length as the response vector. If weights are not given, all weights are taken to equal 1. The default is weights = NULL.
`sc_x`	Logical flag indicating if or not continuous predictors are normalized. The default is sc_x = FALSE.
`sc_y`	Logical flag indicating if or not the response variable is normalized. The default is sc_y = FALSE.
`pen`	User-defined penalty parameter. It must be non-negative. It will only be used in a warped-plane spline fit or a triangle spline fit. The default is pen = 0.
`pnt`	Logical flag indicating if or not penalized constrained regression splines are used. It will only be used in a warped-plane spline fit or a triangle spline fit. The default is pnt = TRUE.
`var.est`	To do a monotonic variance function estimation, the user can set var.est = s.incr(x) or var.est = s.decr(x). See `s.incr` and `s.decr` for more details. The default is var.est = NULL.
`gcv`	Logical flag indicating if or not gcv is used to choose a penalty term in warped-plane surface fit. The default is gcv = FALSE.
`pvf`	Logical flag indicating if or not simulations are used to find the p-value of the test of linear vs double monotone in warped plane surface fit.

Details

We consider generalized partial linear models with independent observations from an exponential family of the form $p(y_i; \theta, \tau) = exp[\{y_i\theta_i - b(\theta_i)\}\tau - c(y_i, \tau)], i = 1, \ldots, n$, where the specifications of the functions b and c determine the sub-family of models. The mean vector $\mu = E(y)$ has values $\mu_i = b'(\theta_i)$, and is related to a design matrix of predictor variables through a monotonically increasing link function $g(\mu_i) = \eta_i, i = 1, \ldots, n$, where η is the systematic component and describes the relationship with the predictors. The relationship between η and θ is determined by the link function b.

For the additive model, the systematic component is specified for each observation by $\eta_i = f_1(x_{1i}) + \ldots + f_L(x_{Li}) + z_i'\beta$, where the functions f_l describe the relationships of the non-parametrically modelled predictors x_l, β is a parameter vector, and z_i contains the values of variables to be modelled parametrically. The non-parametric components are modelled with shape or order assumptions with optional smoothing, and the solution is obtained through an iteratively re-weighted cone projection, with no back-fitting of individual components.

Suppose that η is a n by 1 vector. The matrix form of the systematic component and the predictor is $\eta = \phi_1 + \ldots + \phi_L + Z\beta$, where ϕ_l is the individual component for the lth non-parametrically modelled predictor, $l = 1, \ldots, L$, and Z is an n by p design matrix for the parametrically modelled covariate.

To model the component ϕ_l, smooth regression splines or non-smooth ordinal basis functions can be used. The constraints for the component ϕ_l are in C_l. In the first case, C_l = $\{\phi_l \in R^n : \phi_l = v_l + B_l\beta_l$, where $\beta_l \geq 0$ and $v_l \in V_l\}$, where B_l has regression splines as columns and V_l is the linear space contained in C_l, and in the second case, C_l = $\{\phi \in R^n : A_l\phi \geq 0$ and $B_l\phi = 0\}$, for inequality constraint matrix A_l and equality constraint matrix B_l.

The set C_l is a convex cone and the set $C = C_1 + \ldots + C_p + Z$ is also a convex cone with a finite set of edges, where the edges are the generators of C, and Z is the column space of the design matrix Z for the parametrically modelled covariate.

An iteratively re-weighted cone projection algorithm is used to fit the generalized regression model over the cone C.

See references cited in this section and the official manual (`https://cran.r-project.org/package=coneproj`) for the R package coneproj for more details.

Value

`etahat`	The fitted systematic component η.
`muhat`	The fitted mean value, obtained by transforming the systematic component η by the inverse of the link function.
`vcoefs`	The estimated coefficients for the basis spanning the null space of the constraint set.
`xcoefs`	The estimated coefficients for the edges corresponding to the smooth predictors with no shape constraint and shape-restricted predictors.
`zcoefs`	The estimated coefficients for the parametrically modelled covariate, i.e., the estimation for the vector β.
`ucoefs`	The estimated coefficients for the edges corresponding to the predictors with an umbrella-ordering constraint.

`tcoefs`	The estimated coefficients for the edges corresponding to the predictors with a tree-ordering constraint.
`coefs`	The estimated coefficients for the basis spanning the null space of the constraint set and edges corresponding to the shape-restricted and order-restricted predictors.
`cic`	The cone information criterion proposed in Meyer(2013a). It uses the "null expected degrees of freedom" as a measure of the complexity of the model. See Meyer(2013a) for further details of cic.
`d0`	The dimension of the linear space contained in the cone generated by all constraint conditions.
`edf0`	The estimated "null expected degrees of freedom". It is a measure of the complexity of the model. See Meyer (2013a) and Meyer (2013b) for further details.
`edf`	The constrained effective degrees of freedom.
`etacomps`	The fitted systematic component value for non-parametrically modelled predictors. It is a matrix of which each row is the fitted systematic component value for a non-parametrically modelled predictor. If there are more than one such predictors, the order of the rows is the same as the order that the user defines such predictors in the formula argument of cgam.
`y`	The response variable.
`xmat_add`	A matrix whose columns represent the shape-restricted predictors and smooth predictors with no shape constraint.
`zmat`	A matrix whose columns represent the basis for the parametrically modelled covariate. The user can choose to include a constant vector in it or not. It must have full column rank.
`ztb`	A list keeping track of the order of the parametrically modelled covariate.
`tr`	A matrix whose columns represent the predictors with a tree-ordering constraint.
`umb`	A matrix whose columns represent the predictors with an umbrella-ordering constraint.
`tree.delta`	A matrix whose rows are the edges corresponding to the predictors with a tree-ordering constraint.
`umbrella.delta`	A matrix whose rows are the edges corresponding to the predictors with an umbrella-ordering constraint.
`bigmat`	A matrix whose rows are the basis spanning the null space of the constraint set and the edges corresponding to the shape-restricted and order-restricted predictors.
`shapes`	A vector including the shape and partial-ordering constraints in a cgam fit.
`shapesx`	A vector including the shape constraints in a cgam fit.
`prior.w`	User-defined weights.
`wt`	The weights in the final iteration of the iteratively re-weighted cone projections.
`wt.iter`	Logical flag indicating if or not iteratively re-weighted cone projections may be used. If the response is gaussian, then wt.iter = FALSE; if the response is binomial or poisson, then wt.iter = TRUE.

`family`	The family parameter defined in a cgam formula.
`SSE0`	The sum of squared residuals for the linear part.
`SSE1`	The sum of squared residuals for the full model.
`pvals.beta`	The approximate p-values for the estimation of the vector β. A t-distribution is used as the approximate distribution.
`se.beta`	The standard errors for the estimation of the vector β.
`null_df`	The degree of freedom for the null model of a cgam fit, i.e., the model only containing a constant vector.
`df`	The degree of freedom for the null space of a cgam fit.
`resid_df_obs`	The observed degree of freedom for the residuals of a cgam fit.
`null_deviance`	The deviance for the null model of a cgam fit, i.e., the model only containing a constant vector.
`deviance`	The residual deviance of a cgam fit.
`tms`	The terms objects extracted by the generic function *terms* from a cgam fit.
`capm`	The number of edges corresponding to the shape-restricted predictors.
`capms`	The number of edges corresponding to the smooth predictors with no shape constraint.
`capk`	The number of non-constant columns of zmat.
`capt`	The number of edges corresponding to the tree-ordering predictors.
`capu`	The number of edges corresponding to the umbrella-ordering predictors.
`xid1`	A vector keeping track of the beginning position of the set of edges in bigmat for each shape-restricted predictor and smooth predictor with no shape constraint in xmat.
`xid2`	A vector keeping track of the end position of the set of edges in bigmat for each shape-restricted predictor and smooth predictor with no shape constraint in xmat.
`tid1`	A vector keeping track of the beginning position of the set of edges in bigmat for each tree-ordering factor in tr.
`tid2`	A vector keeping track of the end position of the set of edges in bigmat for each tree-ordering factor in tr.
`uid1`	A vector keeping track of the beginning position of the set of edges in bigmat for each umbrella-ordering factor in umb.
`uid2`	A vector keeping track of the end position of the set of edges in bigmat for each umbrella-ordering factor in umb.
`zid`	A vector keeping track of the positions of the parametrically modelled covariate.
`vals`	A vector storing the levels of each variable used as a factor.
`zid1`	A vector keeping track of the beginning position of the levels of each variable used as a factor.
`zid2`	A vector keeping track of the end position of the levels of each variable used as a factor.

`nsim`	The number of simulations used to get the cic parameter.
`xnms`	A vector storing the names of the shape-restricted predictors and the smooth predictors with no shape constraint in xmat.
`ynm`	The name of the response variable.
`znms`	A vector storing the names of the parametrically modelled covariate.
`is_param`	A logical scalar showing if or not a variable is a parametrically modelled covariate, which could be a linear term or a factor.
`is_fac`	A logical scalar showing if or not a variable is a factor.
`knots`	A list storing the knots used for each shape-restricted predictor and smooth predictor with no shape constraint. For a smooth, constrained and a smooth, unconstrainted predictor, *knots* is a vector of more than 1 elements, and for a shape-restricted predictor without smoothing, *knots* = 0.
`numknots`	A vector storing the number of knots for each shape-restricted predictor and smooth predictor with no shape constraint. For a smooth, constrained and a smooth, unconstrainted predictor, *numknots* > 1, and for a shape-restricted predictor without smoothing, *numknots* = 0.
`sps`	A character vector storing the *space* parameter to create knots for each shape-restricted predictor.
`ms`	The centering terms used to make edges for shape-restricted predictors.
`cpar`	The cpar argument in the cgam formula
`vh`	The estimated monotonic variance function.
`kts.var`	The knots used in monotonic variance function estimation.
`call`	The matched call.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Liao, X. and Meyer, M. C. (2019) cgam: An R Package for the Constrained Generalized Additive Model. *Journal of Statistical Software* ***89(5)***, 1–24.

Meyer, M. C. (2018) A Framework for Estimation and Inference in Generalized Additive Models with Shape and Order Restrictions. *Statistical Science* ***33(4)***, 595–614.

Meyer, M. C. (2013a) Semi-parametric additive constrained regression. *Journal of Nonparametric Statistics* ***25(3)***, 715.

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. and M. Woodroofe (2000) On the degrees of freedom in shape-restricted regression. *Annals of Statistics* ***28***, 1083–1104.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

Mammen, E. and K. Yu (2007) Additive isotonic regression. *IMS Lecture Notes-Monograph Series Asymptotics: Particles, Process, and Inverse Problems* ***55***, 179–195.

Huang, J. (2002) A note on estimating a partly linear model under monotonicity constraints. *Journal of Statistical Planning and Inference* ***107***, 343–351.

Cheng, G.(2009) Semiparametric additive isotonic regression. *Journal of Statistical Planning and Inference* ***139***, 1980–1991.

Bacchetti, P. (1989) Additive isotonic models. *Journal of the American Statistical Association* ***84(405)***, 289–294.

Examples

```
# Example 1.
  data(cubic)
  # extract x
  x <- cubic$x

  # extract y
  y <- cubic$y

  # regress y on x with no restriction with lm()
  fit.lm <- lm(y ~ x + I(x^2) + I(x^3))

  # regress y on x under the restriction: "increasing and convex"
  fit.cgam <- cgam(y ~ incr.conv(x))

  # make a plot to compare the two fits
  par(mar = c(4, 4, 1, 1))
  plot(x, y, cex = .7, xlab = "x", ylab = "y")
  lines(x, fit.cgam$muhat, col = 2, lty = 2)
  lines(x, fitted(fit.lm), col = 1, lty = 1)
  legend("topleft", bty = "n", c("constrained cgam fit", "unconstrained lm fit"),
  lty = c(2, 1), col = c(2, 1))

# Example 2.
  library(MASS)
  data(Rubber)

  # regress loss on hard and tens under the restrictions:
  # "decreasing" and "decreasing"
  fit.cgam <- cgam(loss ~ decr(hard) + decr(tens), data = Rubber)
  # "smooth and decreasing" and "smooth and decreasing"
  fit.cgam.s <- cgam(loss ~ s.decr(hard) + s.decr(tens), data = Rubber)
  summary(fit.cgam.s)
  anova(fit.cgam.s)
  # smooth components plot
  plot(fit.cgam.s, pages = 1)

  # make a 3D plot based on fit.cgam and fit.cgam.s
  ctl <- list(th = 120, main = "3D Plot of a Cgam Fit")
  plotpersp(fit.cgam, control = ctl)
  ctl <- list(th = 120, main = "3D Plot of a Smooth Cgam Fit")
```

```
  plotpersp(fit.cgam.s, "tens", "hard", control = ctl)

# Example 3. monotonic variance estimation
  n <- 400
  x <- runif(n)
  sig <- .1 + exp(15*x-8)/(1+exp(15*x-8))
  e <- rnorm(n)
  mu <- 10*x^2
  y <- mu + sig*e

  fit <- cgam(y ~ s.incr.conv(x), var.est = s.incr(x))
  est.var <- fit$vh
  muhat <- fit$muhat

  par(mfrow = c(1, 2))
  plot(x, y)
  points(sort(x), muhat[order(x)], type = "l", lwd = 2, col = 2)
  lines(sort(x), (mu)[order(x)], col = 4)

  plot(sort(x), est.var[order(x)], col=2, lwd=2, type="l",
  lty=2, ylab="Variance", ylim=c(0, max(c(est.var, sig^2))))
  points(sort(x), (sig^2)[order(x)], col=1, lwd=2, type="l")

## Not run:
# Example 4. monotonic variance estimation with the lidar data set in SemiPar
  library(SemiPar)
  data(lidar)

  fit <- cgam(logratio ~ s.decr(range), var.est=s.incr(range), data=lidar)
  muhat <- fit$muhat
  est.var <- fit$vh

  logratio <- lidar$logratio
  range <- lidar$range
 pfit <- predict(fit, newData=data.frame(range=range), interval="confidence", level=0.95)
  upp <- pfit$upper
  low <- pfit$lower

  par(mfrow = c(1, 2))
  plot(range, logratio)
  points(sort(range), muhat[order(range)], type = "l", lwd = 2, col = 2)
  lines(sort(range), upp[order(range)], type = "l", lwd = 2, col = 4)
  lines(sort(range), low[order(range)], type = "l", lwd = 2, col = 4)
  title("Smoothly Decreasing Fit with a Point-Wise Confidence Interval", cex.main=0.5)

  plot(range, est.var, col=2, lwd=2, type="l",lty=2, ylab="variance")
  title("Smoothly Increasing Variance", cex.main=0.5)

## End(Not run)
```

`cgamm` *Constrained Generalized Additive Mixed-Effects Model Fitting*

Description

This routine is an addition to the main routine cgam in this package. A random-intercept component is included in a cgam model.

Usage

```
cgamm(formula, nsim = 100, family = gaussian(), cpar = 1.2, data = NULL, weights = NULL,
sc_x = FALSE, sc_y = FALSE, bisect = TRUE, reml = TRUE,
multicore = getOption("cgam.parallel"), CIC = FALSE, nAGQ = 1L)
```

Arguments

`formula`	A formula object which gives a symbolic description of the model to be fitted. It has the form "response ~ predictor + (1|id)", where id is the label for a group effect. For now, only gaussian responses are considered and this routine only includes a random-intercept effect. See `cgam` for more details.
`nsim`	The number of simulations used to get the cic parameter. The default is nsim = 0.
`family`	A parameter indicating the error distribution and link function to be used in the model. For now, the options are family = gaussian() and family = binomial().
`cpar`	A multiplier to estimate the model variance, which is defined as $\sigma^2 = SSR/(n - cpar * edf)$. SSR is the sum of squared residuals for the full model and edf is the effective degrees of freedom. The default is cpar = 1.2. The user-defined value must be between 1 and 2. See Meyer, M. C. and M. Woodroofe (2000) for more details.
`data`	An optional data frame, list or environment containing the variables in the model. The default is data = NULL.
`weights`	An optional non-negative vector of "replicate weights" which has the same length as the response vector. If weights are not given, all weights are taken to equal 1. The default is weights = NULL.
`sc_x`	Logical flag indicating if or not continuous predictors are normalized. The default is sc_x = FALSE.
`sc_y`	Logical flag indicating if or not the response variable is normalized. The default is sc_y = FALSE.
`bisect`	If bisect = TRUE, a 95 percent confidence interval will be found for the variance ratio parameter by a bisection method.
`reml`	If reml = TRUE, restricted maximum likelihood (REML) method will be used to find estimates instead of maximum likelihood estimation (MLE).
`multicore`	A parameter retrieving the current global option for "csurvey.multicore" and assigns its value to the multicore variable, allowing the function to respect a user-defined setting for parallel processing behavior across the package.
`CIC`	Logical flag indicating if or not cic is computed.
`nAGQ`	Integer scalar - the number of points per axis for evaluating the adaptive Gauss-Hermite approximation to the log-likelihood. Defaults to 1, corresponding to the Laplace approximation. Values greater than 1 produce greater accuracy in the evaluation of the log-likelihood at the expense of speed.

Value

`muhat`	The fitted fixed-effect term.
`ahat`	A vector of estimated random-effect terms.
`sig2hat`	Estimate of the variance (σ^2) of between-cluster error terms.
`siga2hat`	Estimate of the variance (σ_a^2) of within-cluster error terms.
`thhat`	Estimate of the ratio (θ) of two variances.
`pv.siga2`	p-value of the test $H_0 : \sigma_a^2 = 0$
`ci.siga2`	95 percent confidence interval for the variance of within-cluster error terms.
`ci.th`	95 percent confidence interval for ratio of two variances.
`ci.rho`	95 percent confidence interval for intraclass correlation coefficient.
`ci.sig2`	95 percent confidence interval for the variance of between-cluster error terms.
`call`	The matched call.

Author(s)

Xiyue Liao

Examples

```
# Example 1 (family = gaussian).

# simulate a balanced data set with 30 clusters
# each cluster has 30 data points
n <- 30
m <- 30

# the standard deviation of between cluster error terms is 1
# the standard deviation of within cluster error terms is 2
sige <- 1
siga <- 2

# generate a continuous predictor
x <- 1:(m*n)
for(i in 1:m) {
x[(n*(i-1)+1):(n*i)] <- round(runif(n), 3)
}
# generate a group factor
group <- trunc(0:((m*n)-1)/n)+1

# generate the fixed-effect term
mu <- 10*exp(10*x-5)/(1+exp(10*x-5))

# generate the random-intercept term asscosiated with each group
avals <- rnorm(m, 0, siga)

# generate the response
y <- 1:(m*n)
for(i in 1:m){
```

```
y[group == i] <- mu[group == i] + avals[i] + rnorm(n, 0, sige)
}

# use REML method to fit the model
ans <- cgamm(y ~ s.incr(x) + (1|group), reml=TRUE)
summary(ans)
anova(ans)

muhat <- ans$muhat
plot(x, y, col = group, cex = .6)
lines(sort(x), mu[order(x)], lwd = 2)
lines(sort(x), muhat[order(x)], col = 2, lty = 2, lwd = 2)
legend("topleft", bty = "n", c("true fixed-effect term", "cgamm fit"),
col = c(1, 2), lty = c(1, 2), lwd = c(2, 2))

# Example 2 (family = binomial).
# simulate a balanced data set with 20 clusters
# each cluster has 20 data points

  n <- 20
  m <- 20#
  N <- n*m

  # siga is the sd for the random intercept
  siga <- 1

# generate a group factor
  group <- trunc(0:((m*n)-1)/n)+1
  group <- factor(group)

# generate the random-intercept term asscosiated with each group
  avals <- rnorm(m,0,siga)

# generate the fixed-effect mean term: mu, systematic term: eta and the response: y
  x <- runif(m*n)
  mu <- 1:(m*n)
  y <- 1:(m*n)

  eta <- 2 * (1 + tanh(7 * (x - .8))) - 2
  eta0 <- eta
  for(i in 1:m){eta[group == i] <- eta[group == i] + avals[i]}
  for(i in 1:m){mu[group == i] <- 1 - 1 / (1 + exp(eta[group == i]))}
  for(i in 1:m){y[group == i] <- rbinom(n, size = 1, prob = mu[group == i])}
  dat <- data.frame(x = x, y = y, group = group)
  ansc <- cgamm(y ~ s.incr.conv(x) + (1|group),
  family = binomial(link = "logit"), reml = FALSE, data = dat)
  summary(ansc)
  anova(ansc)
```

`COforest` *Colorado Forest Data Set*

Description

This data set contains 9167 records of different species of live trees for 345 sampled forested plots measured in 2015.

Usage

```
data("COforest")
```

Format

A data frame with 9167 observations on the following 19 variables.

`PLT_CN` Unique identifier of plot

`STATECD` State code using Bureau of Census Federal Information Processing Standards (FIPS)

`COUNTYCD` County code (FIPS)

`ELEV_PUBLIC` Elevation (ft) extracted spatially using LON_PUBLIC/LAT_PUBLIC

`LON_PUBLIC` Fuzzed longitude in decimal degrees using NAD83 datum

`LAT_PUBLIC` Fuzzed latitude in decimal degrees using NAD83 datum

`ASPECT` a numeric vector

`SLOPE` a numeric vector

`SUBP` Subplot number

`TREE` Tree number within subplot

`STATUSCD` Tree status (0:no status; 1:live tree; 2:dead tree; 3:removed)

`SPCD` Species code

`DIA` Current diameter (in)

`HT` Total height (ft): estimated when broken or missing

`ACTUALHT` Actual height (ft): measured standing or down

`HTCD` Height method code (1:measured; 2:actual measured-length estimated; 3:actual and length estimated; 4:modeled

`TREECLCD` Tree class code (2:growing-stock; 3:rough cull; 4:rotten cull)

`CR` Compacted crown ratio (percentage)

`CCLCD` Crown class (1:open grown; 2:domimant; 3:co-dominant; 4:intermediate; 5:overtopped)

Source

It is provided by Forest Inventory Analysis (FIA) National Program.

References

X. Liao and M. Meyer (2019). Estimation and Inference in Mixed-Effect Regression Models using Shape Constraints, with Application to Tree Height Estimation. (*to appear in Journal of the Royal Statistical Society. Series C: Applied Statistics*)

Examples

```
## Not run:
  library(dplyr)
  library(tidyr)
  data(COforest)

  #re-grouping classes of CCLCD:
  #combine dominant (2) and co-dominant (3)
  #combine intermediate (4) and overtopped (5)
  COforest = COforest |> mutate(CCLCD = replace(CCLCD, CCLCD == 2, 3)) |>
    mutate(CCLCD = replace(CCLCD, CCLCD == 5, 4))

  #make a list of species, each element is a small data frame for one species
  species = COforest |> group_by(SPCD) |> nest()

  #get the subset for quaking aspen, which is the 4th element in the species list
  sub = species$data[[4]]
  #for quaking aspen, there are only two crown classes: dominant/co-dominant
  #and intermediate/overtopped
  table(sub$CCLCD)
  #   3    4
  #1400  217
  #for quaking aspen, there are only two tree clases: growing-stock and rough cull
  table(sub$TREECLCD)
  #2    3
  #1591   26

  #fit the model
  ansc = cgamm(log(HT)~s.incr.conc(DIA)+factor(CCLCD)+factor(TREECLCD)
  +(1|PLT_CN), reml=TRUE, data=sub)

  #check which classes are significant
  summary(ansc)

  #fixed-effect 95
  newData = data.frame(DIA=sub$DIA,CCLCD=sub$CCLCD,TREECLCD=sub$TREECLCD)
  pfit = predict(ansc, newData,interval='confidence')
  lower = pfit$lower
  upper = pfit$upper
  #we need to use exp(muhat) later in the plot
  muhat = pfit$fit

  #get x and y
  x = sub$DIA
  y = sub$HT

  #get TREECLCD and CCLCD
  z1 = sub$TREECLCD
  z2 = sub$CCLCD

  #plot fixed-effect confidence intervals
  plot(x, y, xlab='DIA (m)', ylab='HT (m)', ylim=c(min(y),max(exp(upper))+10),type='n')
```

```
  lines(sort(x[z2==3&z1==2]), (exp(pfit$fit)[z2==3&z1==2])[order(x[z2==3&z1==2])],
  col='slategrey', lty=1, lwd=2)
  lines(sort(x[z2==3&z1==2]), (exp(pfit$lower)[z2==3&z1==2])[order(x[z2==3&z1==2])],
  col='slategrey', lty=1, lwd=2)
  lines(sort(x[z2==3&z1==2]), (exp(pfit$upper)[z2==3&z1==2])[order(x[z2==3&z1==2])],
  col='slategrey', lty=1, lwd=2)

  lines(sort(x[z2==4&z1==2]), (exp(pfit$fit)[z2==4&z1==2])[order(x[z2==4&z1==2])],
  col=”blueviolet”, lty=2, lwd=2)
  lines(sort(x[z2==4&z1==2]), (exp(pfit$lower)[z2==4&z1==2])[order(x[z2==4&z1==2])],
  col=”blueviolet”, lty=2, lwd=2)
  lines(sort(x[Cz2==4&z1==2]), (exp(pfit$upper)[Cz2==4&z1==2])[order(x[Cz2==4&z1==2])],
  col=”blueviolet”, lty=2, lwd=2)

  lines(sort(x[Cz2==3&z1==3]), (exp(pfit$fit)[Cz2==3&z1==3])[order(x[Cz2==3&z1==3])],
  col=3, lty=3, lwd=2)
  lines(sort(x[Cz2==3&z1==3]), (exp(pfit$lower)[Cz2==3&z1==3])[order(x[Cz2==3&z1==3])],
  col=3, lty=3, lwd=2)
  lines(sort(x[Cz2==3&z1==3]), (exp(pfit$upper)[Cz2==3&z1==3])[order(x[Cz2==3&z1==3])],
  col=3, lty=3, lwd=2)

  lines(sort(x[Cz2==4&z1==3]), (exp(pfit$fit)[Cz2==4&z1==3])[order(x[Cz2==4&z1==3])],
  col=2, lty=4, lwd=2)
  lines(sort(x[Cz2==4&z1==3]), (exp(pfit$lower)[Cz2==4&z1==3])[order(x[Cz2==4&z1==3])],
  col=2, lty=4, lwd=2)
  lines(sort(x[Cz2==4&z1==3]), (exp(pfit$upper)[Cz2==4&z1==3])[order(x[Cz2==4&z1==3])],
  col=2, lty=4, lwd=2)
  legend('bottomright', bty='n', cex=.9,c('dominant/co-dominant and growing stock',
  'intermediate/overtopped and growing stock','dominant/co-dominant and rough cull',
  'intermediate/overtopped and rough cull'), col=c('slategrey',”blueviolet”,3,2),
  lty=c(1,2,3,4),lwd=c(2,2,2,2), pch=c(24,23,22,21))
  title('Quaking Aspen fits with 95

## End(Not run)
```

`conc` *Specify a Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is concave in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
conc(x, numknots = 0, knots = 0, space = ”E”)
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"conc" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "conc"; the shape attribute is 4("concave"), and according to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be concave, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 4("concave"); numknots: the numknots argument in "conc"; knots: the knots argument in "conc"; space: the space argument in "conc".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`conv`

Examples

```
# generate y
x <- seq(-1, 2, by = 0.1)
n <- length(x)
y <- - x^2 + rnorm(n, .3)
```

```
# regress y on x under the shape-restriction: "concave"
ans <- cgam(y ~ conc(x))

# make a plot
plot(x, y)
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "concave fit", col = 2, lty = 1)
```

`conv` *Specify a Convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is convex in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
conv(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"conv" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "conv"; the shape attribute is 3("convex"), and according to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be convex, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 3("convex"); numknots: the numknots argument in "conv"; knots: the knots argument in "conv" ; space: the space argument in "conv".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`conc`

Examples

```
# generate y
x <- seq(-1, 2, by = 0.1)
n <- length(x)
y <- x^2 + rnorm(n, .3)

# regress y on x under the shape-restriction: "convex"
ans <- cgam(y ~ conv(x))

# make a plot
plot(x, y)
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "convex fit", col = 2, lty = 1)
```

`cubic` *A Data Set for Cgam*

Description

This data set is used for several examples in the cgam package.

Usage

`data(cubic)`

Format

A data frame with 50 observations on the following 2 variables.

`x` The predictor vector.

`y` The response vector.

Source

STAT640 HW 14 given by Dr. Meyer.

---

decr *Specify a Decreasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is decreasing in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
decr(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"decr" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "decr"; the shape attribute is 2("decreasing"), and according to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be decreasing, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "decr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 2("decreasing"); numknots: the numknots argument in "decr"; knots: the knots argument in "decr"; space: the space argument in "decr".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`decr.conc`, `decr.conv`

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "decreasing"
ans <- cgam(y ~ decr(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("bottomright", bty = "n", "decreasing fit", col = 2, lty = 1)
```

`decr.conc` *Specify a Decreasing and Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is decreasing and concave in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
decr.conc(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"decr.conc" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "decr.conc"; the shape attribute is 8("decreasing and concave"), and according to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be decreasing and concave, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "decr.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 8("decreasing and concave"); numknots: the numknots argument in "decr.conc"; knots: the knots argument in "decr.conc"; space: the space argument in "decr.conc".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`decr.conv`, `decr`

Examples

```
data(cubic)

# extract x
x <-  cubic$x
```

```
# extract y
y <- - cubic$y

# regress y on x with the shape restriction: "decreasing" and "concave"
ans <- cgam(y ~ decr.conc(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "decreasing and concave fit", col = 2, lty = 1)
```

`decr.conv` *Specify a Decreasing and Convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is decreasing and convex in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
decr.conv(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"decr.conv" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "decr.conv"; the shape attribute is 6("decreasing and convex"), and according to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be decreasing and convex, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "decr.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 6("decreasing and convex"); numknots: the numknots argument in "decr.conv"; knots: the knots argument in "decr.conv"; space: the space argument in "decr.conv".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`decr.conc`, `decr`

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "decreasing" and "convex"
ans <- cgam(y ~ decr.conv(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("bottomright", bty = "n", "decreasing and convex fit", col = 2, lty = 1)
```

incr *Specify an Increasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is increasing in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
incr(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"incr" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "incr"; the shape attribute is 1("increasing"), and according to the value of the vector itself and its attributes, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be increasing, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "incr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 1("increasing"); numknots: the numknots argument in "incr"; knots: the knots argument in "incr"; space: the space argument in "incr".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`incr.conc`, `incr.conv`

Examples

```
data(cubic)

# extract x
x <- cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "increasing"
ans <- cgam(y ~ incr(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "increasing fit", col = 2, lty = 1)
```

`incr.conc` *Specify an Increasing and Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is increasing and concave in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
incr.conc(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"incr.conc" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "incr.conc"; the shape attribute is 7("increasing and concave"), and according to the value of the vector itself and its attributes, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be increasing and concave, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "incr.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 7("increasing and concave"); numknots: the numknots argument in "incr.conc"; knots: the knots argument in "incr.conc"; space: the space argument in "incr.conc".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`incr.conv`

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- - cubic$y

# regress y on x with the shape restriction: "increasing" and "concave"
ans <- cgam(y ~ incr.conc(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
```

```
legend("topleft", bty = "n", "increasing and concave fit", col = 2, lty = 1)
```

`incr.conv` *Specify an Increasing and Convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is increasing and convex in a predictor in a formula argument to cgam. This is the unsmoothed version.

Usage

```
incr.conv(x, numknots = 0, knots = 0, space = "E")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`knots`	The knots used to smoothly constrain a predictor. The value should be 0 for a shape-restricted predictor without smoothing. The default value is 0.
`space`	A character specifying the method to create knots. It will not be used for a shape-restricted predictor without smoothing. The default value is "E".

Details

"incr.conv" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "incr.conv"; the shape attribute is 5("increasing and convex"), and according to the value of the vector itself and its attributes, the cone edges of the cone generated by the constraint matrix, which constrains the relationship between the systematic component η and "x" to be increasing and convex, will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "incr.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 5("increasing and convex"); numknots: the numknots argument in "incr.conv"; knots: the knots argument in "incr.conv"; space: the space argument in "incr.conv".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`incr.conc`, `incr`

Examples

```
data(cubic)

# extract x
x <- cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "increasing" and "convex"
ans <- cgam(y ~ incr.conv(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "increasing and convex fit", col = 2, lty = 1)
```

`mental` *Alachua County Study of Mental Impairment*

Description

The date set is from a study of mental health for a random sample of 40 adult residents of Alachua County, Florida. Mental impairment is an ordinal response with 4 categories: well, mild symptom formation, moderate symptom formation, and impaired, which are recorded as 1, 2, 3, and 4. Life event index is a composite measure of the number and severity of important life events that occurred with the past three years, e.g., birth of a child, new job, divorce, or death of a family member. It is an integer from 0 to 9. Another covariate is socio-economic status and it is measured as binary: high = 1, low = 0.

Usage

```
data(mental)
```

Format

`mental` Mental impairment. It is an ordinal response with 4 categories recorded as 1, 2, 3, and 4.

`ses` Socio-economic status measured as binary: high = 1, low = 0.

`life` Life event index. It is an integer from 0 to 9.

References

Agresti, A. (2010) Analysis of Ordinal Categorical Data, *2nd ed. Hoboken, NJ: Wiley*.

See Also

`Ord`

Examples

```
# proportional odds model example
data(mental)

# model the relationship between the latent variable and life event index as increasing
# socio-economic status is included as a binary covariate
fit.incr <- cgam(mental ~ incr(life) + ses, data = mental, family = Ord)

# check the estimated probabilities P(mental = k), k = 1, 2, 3, 4
probs.incr <- fitted(fit.incr)
head(probs.incr)
```

`Ord` *Specify an Ordered Categorical Family in a CGAM Formula*

Description

This is a subroutine to specify an ordered catergorical family in a cgam formula. It set things up to a routine called cgam.polr. This is learned from the polr routine in the MASS package, which fits a logistic or probit regression model to an ordered categorical response. Currently only the logistic regression model is allowed.

Usage

```
Ord(link = "identity")
```

Arguments

`link`	The link function. Users don't need specify this term.

Details

See the polr section in the official manual of the MASS package (`https://cran.r-project.org/package=MASS`) for details.

Value

`muhat`	The estimated expected value of a latent variable.
`zeta`	Estimated cut-points defining the intervals of a latent variable such that the latent variable is between two adjacent cut-points is equivalent to that the ordered categorical response is in a category.

Author(s)

Xiyue Liao

References

Agresti, A. (2002) *Categorical Data*. Second edition. Wiley.

See Also

`mental`

Examples

```
## Not run:
# Example 1.
# generate the predictor and the latenet variable
n <- 500
set.seed(123)
x <- runif(n, 0, 1)
yst <- 5*x^2 + rlogis(n)

# generate observed ordered response, which has levels 1, 2, 3.
cts <- quantile(yst, probs = seq(0, 1, length = 4))
yord <- cut(yst, breaks = cts, include.lowest = TRUE, labels = c(1:3), Ord = TRUE)
y <- as.numeric(levels(yord))[yord]

# regress y on x under the shape-restriction: the latent variable is "increasing-convex"
# w.r.t x
ans <- cgam(y ~ s.incr.conv(x), family = Ord)

# check the estimated cut-points
ans$zeta

# check the estimated expected value of the latent variable
head(ans$muhat)

# check the estimated probabilities P(y = k), k = 1, 2, 3
head(fitted(ans))

# check the estimated latent variable
plot(x, yst, cex = 1, type = "n", ylab = "latent variable")
cols <- topo.colors(3)
for (i in 1:3) {
points(x[y == i], yst[y == i], col = cols[i], pch = i, cex = 0.7)
```

```
}
for (i in 1:2) {
abline(h = (ans$zeta)[i], lty = 4, lwd = 1)
}
lines(sort(x), (5*x^2)[order(x)], lwd = 2)
lines(sort(x), (ans$muhat)[order(x)], col = 2, lty = 2, lwd = 2)
legend("topleft", bty = "n", col = c(1, 2), lty = c(1, 2),
c("true latent variable", "increasing-convex fit"), lwd = c(1, 1))

## End(Not run)
## Not run:
# Example 2. mental impairment data set
# mental impairment is an ordinal response with 4 categories recorded as 1, 2, 3, and 4
# two covariates are life event index and socio-economic status (high = 1, low = 0)
data(mental)
table(mental$mental)

# model the relationship between the latent variable and life event index as increasing
# socio-economic status is included as a binary covariate
fit.incr <- cgam(mental ~ incr(life) + ses, data = mental, family = Ord)

# check the estimated probabilities P(mental = k), k = 1, 2, 3, 4
probs.incr <- fitted(fit.incr)
head(probs.incr)

## End(Not run)
```

`plasma` *A Data Set for Cgam*

Description

This data set is used for the routine plotpersp. It contains 314 observations of blood plasma, beta carotene measurements along with several covariates. High levels of blood plasma and beta carotene are believed to be protective against cancer, and it is of interest to determine the relationships with covariates.

Usage

```
data(plasma)
```

Format

`logplasma` A numeric vector of the logarithm of plasma levels.

`betacaro` A numeric vector of dietary beta carotene consumed mcg per day.

`bmi` A numeric vector of BMI values.

`cholest` A numeric vector of cholesterol consumed mg per day.

`dietfat` A numeric vector of the logarithm of grams of diet fat consumed per day.

`fiber` A numeric vector of grams of fiber consumed per day.

`retinol` A numeric vector of retinol consumed per day.

`smoke` A numeric vector of smoking status (1=Never, 2=Former, 3=Current Smoker).

`vituse` A numeric vector of vitamin use (1=Yes, fairly often, 2=Yes, not often, 3=No).

References

Nierenberg, D.,Stukel, T.,Baron, J.,Dain, B.,and Greenberg, E. (1989) Determinants of plasma levels of beta-carotene and retinol. *American Journal of Epidemiology* ***130***, 511–521.

Examples

```
data(plasma)
```

`plotpersp` *Create a 3D Plot for a CGAM Object*

Description

Given an object of the cgam class, which has at least two non-parametrically modelled predictors, this routine will make a 3D plot of the fit with a set of two non-parametrically modelled predictors in the formula being the x and y labs. If there are more than two non-parametrically modelled predictors, any other such predictor will be evaluated at the largest value which is smaller than or equal to its median value.

If there is any categorical covariate and if the user specifies the argument categ to be a character representing a categorical covariate in the formula, then a 3D plot with multiple parallel surfaces, which represent the levels of a categorical covariate in an ascending order, will be created; otherwise, a 3D plot with only one surface will be created. Each level of a categorical covariate will be evaluated at its mode.

This routine is extended to make a 3D plot for an object fitted by warped-plane splines or triangle splines. Note that two non-parametrically modelled predictors specified in this routine must both be modelled as addtive components, or a pair of predictors forming an isotonic or convex surface without additivity assumption.

This routine is an extension of the generic R graphics routine persp. See the documentation below for more details.

Usage

```
plotpersp(object,...)
```

Arguments

`object`	An object of the cgam class with at least two non-parametrically modelled predictors.
`...`	Additional arguments passed to the plotting method, such as `x1`, `x2`, and `control`. These include: `x1` Optional; either a character string or a vector matching the first nonparametric predictor. If omitted, the first predictor's name is used. `x2` Optional; either a character string or a vector matching the second nonparametric predictor. If omitted, the second predictor's name is used. `control` A list of graphical control parameters, typically created with `plotpersp_control`. Controls aspects such as colors, axis limits, rotation angles, and grid resolution.

Value

The routine plotpersp returns a 3D plot of an object of the cgam class. The x lab and y lab represent a set of non-parametrically modelled predictors used in a cgam formula, and the z lab represents the estimated mean value or the estimated systematic component value.

Author(s)

Mary C. Meyer and Xiyue Liao

Examples

```
# Example 1.
  data(FEV)

  # extract the variables
  y <- FEV$FEV
  age <- FEV$age
  height <- FEV$height
  sex <- FEV$sex
  smoke <- FEV$smoke

  fit <- cgam(y ~ incr(age) + incr(height) + factor(sex) + factor(smoke), nsim = 0)
  fit.s <- cgam(y ~ s.incr(age) + s.incr(height) + factor(sex) + factor(smoke), nsim = 0)

  ctl <- list(ngrid = 10, main = "Cgam Increasing Fit",
  sub = "Categorical Variable: Sex", categ = "factor(sex)")
  plotpersp(fit, age, height, control = ctl)

  ctl <- list(ngrid = 10, main = "Cgam Smooth Increasing Fit",
  sub = "Categorical Variable: Smoke", categ = "factor(smoke)")
  plotpersp(fit.s, age, height, control = ctl)

# Example 2.
  data(plasma)

 fit <- cgam(logplasma ~  s.decr(bmi) + s.decr(logdietfat) + s.decr(cholest) + s.incr(fiber)
```

```
+ s.incr(betacaro) + s.incr(retinol) + factor(smoke) + factor(vituse), data = plasma)

  ctl <- list(ngrid = 15, th = 120, ylab = "log(dietfat)",
            zlab = "est mean of log(plasma)", main = "Cgam Fit with the Plasma Data Set",
             sub = "Categorical Variable: Vitamin Use", categ = "factor(vituse)")

 plotpersp(fit, "bmi", "logdietfat", control = ctl)

# Example 3.
  data(plasma)
  addl <- 1:314*0 + 1
  addl[runif(314) < .3] <- 2
  addl[runif(314) > .8] <- 4
  addl[runif(314) > .8] <- 3

  plasma$addl <- addl
  ans <- cgam(logplasma ~ s.incr(betacaro, 5) + s.decr(bmi) +
              s.decr(logdietfat) + as.factor(addl), data = plasma)

  ctl <- list(th = 240, random = TRUE, categ = "as.factor(addl)")
  plotpersp(ans, "betacaro", "logdietfat", control = ctl)

# Example 4.
## Not run:
  n <- 100
  set.seed(123)
  x1 <- sort(runif(n))
  x2 <- sort(runif(n))
  y <- 4 * (x1 - x2) + rnorm(n, sd = .5)

  # regress y on x1 and x2 under the shape-restriction: "decreasing-increasing"
  # with a penalty term = .1
  ans <- cgam(y ~ s.decr.incr(x1, x2), pen = .1)

# plot the constrained surface
  plotpersp(ans)

## End(Not run)
```

`plotpersp_control` *Control Parameters for plotpersp.cgam*

Description

Constructs a list of control parameters to customize the appearance of perspective plots created by `plotpersp.cgam`.

Usage

```
plotpersp_control(surface = "mu", x1nm = NULL, x2nm = NULL, categ = NULL,
```

```
  col = NULL, random = FALSE, ngrid = 12, xlim = NULL, ylim = NULL, zlim = NULL,
  xlab = NULL, ylab = NULL, zlab = NULL, th = NULL, ltheta = NULL,
  main = NULL, sub = NULL, ticktype = "simple")
```

Arguments

`surface`	Character string indicating the surface type to plot. Options are `"mu"` for fitted means or `"eta"` for systematic component.
`x1nm`	Optional character strings specifying names of the predictors for the x axis.
`x2nm`	Optional character strings specifying names of the predictors for the y axis.
`categ`	Optional character string naming a categorical covariate in the `cgam` fit to stratify surfaces. Default is `NULL`.
`col`	Colors used for the surface(s). If `NULL`, defaults are applied automatically depending on the number of surfaces.
`random`	Logical; if `TRUE`, colors are randomized for multiple surfaces. Default is `FALSE`.
`ngrid`	Integer; number of grid points along each axis. Default is `12`.
`xlim`, `ylim`, `zlim`	Optional numeric vectors of length 2 specifying limits for the x, y, and z axes.
`xlab`, `ylab`, `zlab`	Axis labels for x, y, and z axes. Default is based on the predictor names and surface type.
`th`	Angle defining the azimuthal direction for viewing the plot (theta).
`ltheta`	Angle defining the direction of lighting (light theta).
`main`	Main title of the plot.
`sub`	Subtitle of the plot.
`ticktype`	Character string indicating type of tick marks: `"simple"` (default) or `"detailed"`.

Value

A named list of control settings for use with `plotpersp`.

Author(s)

Mary C. Meyer and Xiyue Liao

See Also

`plotpersp`, `persp`

Examples

```
ctrl <- plotpersp_control(col = "topo", ngrid = 20, th = 45)
# Then used inside plotpersp:
# plotpersp(fit, control = ctrl)
```

`predict.cgam` *Predict Method for CGAM Fits*

Description

Predicted values based on a cgam object

Usage

```
## S3 method for class 'cgam'
predict(object, newdata, interval = c("none", "confidence", "prediction"),
type = c("response", "link"), level = 0.95, n.mix = 500,...)
```

Arguments

`object`	A cgam object.
`newdata`	A data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`interval`	Type of interval calculation. A prediction interval is only implemented for Gaussian response for now.
`type`	If the response is Gaussian, type = "response" gives the predicted mean; if the response is binomial, type = "response" gives the predicted probabilities, and type = "link" gives the predicted systematic component.
`level`	Tolerance/confidence level.
`n.mix`	Number of simulations to get the mixture distribution. The default is n.mix = 500.
`...`	Further arguments passed to the routine.

Details

Constrained spline estimators can be characterized by projections onto a polyhedral convex cone. Point-wise confidence intervals for constrained splines are constructed by estimating the probabilities that the projection lands on each of the faces of the cone, and using a mixture of covariance matrices to estimate the standard error of the function estimator at any design point.

Note that currently predict.cgam only works when all components in a cgam formula are additive.

See references cited in this section for more details.

Value

`fit`	A vector of predictions.
`lower`	A vector of lower bound if interval is set to be "confidence".
`upper`	A vector of upper bound if interval is set to be "confidence".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Constrained partial linear regression splines. *Statistical Sinica in press*.

Meyer, M. C. (2017) Confidence intervals for regression functions using constrained splines with application to estimation of tree height

Meyer, M. C. (2012) Constrained penalized splines. *Canadian Journal of Statistics* ***40(1)***, 190–206.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

Examples

```
# Example 1.
# generate data
n <- 100
set.seed(123)
x <- runif(n)
y <- 4*x^3 + rnorm(n)

# regress y on x under the shape-restriction: "increasing-convex"
fit <- cgam(y ~ s.incr.conv(x))

# make a data frame
x0 <- seq(min(x), max(x), by = 0.05)

# predict values in new.Data based on the cgam fit without a confidence interval
pfit <- predict(fit, newdata = data.frame(x = x0))

# make a plot to check the prediction
plot(x, y, main = "Predict Method for CGAM")
lines(sort(x), (fitted(fit)[order(x)]))
points(x0, pfit$fit, col = 2, pch = 20)

# predict values in newdata based on the cgam fit with a 95 percent confidence interval
pfit <- predict(fit, newdata = data.frame(x = x0), interval = "confidence", level = 0.95)

# make a plot to check the prediction
plot(x, y, main = "Pointwise Confidence Bands (Gaussian Response)")
lines(sort(x), (fitted(fit)[order(x)]))
lines(sort(x0), (pfit$lower)[order(x0)], col = 2, lty = 2)
lines(sort(x0), (pfit$upper)[order(x0)], col = 2, lty = 2)
points(x0, pfit$fit, col = 2, pch = 20)

# Example 2. binomial response
n <- 200
x <- seq(0, 1, length = n)

eta <- 4*x^2 - 2
```

```
mu <- exp(eta)/(1+exp(eta))
set.seed(123)
y <- 1:n*0
y[runif(n)<mu] = 1

fit <- cgam(y ~ s.incr.conv(x), family = binomial)
muhat <- fitted(fit)

  # predict values in new.Data based on the cgam fit with a 95 percent confidence interval
  xinterp <- seq(min(x), max(x), by = 0.05)
pfit <- predict(fit, newdata = data.frame(x = xinterp),
  interval = "confidence", level = 0.95)
pmu <- pfit$fit
lwr <- pfit$lower
upp <- pfit$upper

# make a plot to check the prediction
plot(x, y, type = "n", ylim = c(0, 1),
main = "Pointwise Confidence Bands (Binomial Response)")
rug(x[y == 0])
    rug(x[y == 1], side = 3)
lines(x, mu)
lines(x, muhat, col = 5, lty = 2)

points(xinterp, pmu, pch = 20)
lines(xinterp, upp, col = 5)
points(xinterp, upp, pch = 20)
lines(xinterp, lwr, col = 5)
points(xinterp, lwr, pch = 20)
```

`s` *Specify a Smooth Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.

`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s"; the shape attribute is 17("smooth"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 17("smooth"); numknots: the numknots argument in "s"; knots: the knots argument in "s"; space: the space argument in "s".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`s.incr`, `s.decr`, `s.conc`, `s.conv`, `s.incr.conc`, `s.incr.conv`, `s.decr.conc`, `s.decr.conv`

Examples

```
# generate y
x <- seq(-1, 2, by = 0.1)
n <- length(x)
y <- - x^2 + rnorm(n, .3)

# regress y on x under the shape-restriction: "smooth"
ans <- cgam(y ~ s(x))
knots <- ans$knots[[1]]
```

```
# make a plot
plot(x, y)
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth fit", col = 2, lty = 1)
legend(1.6, 1.8, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.conc` *Specify a Smooth and Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth and concave in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.conc(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s.conc" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.conc"; the shape attribute is 12("smooth and concave"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 12("smooth and concave"); numknots: the numknots argument in "s.conc"; knots: the knots argument in "s.conc"; space: the space argument in "s.conc".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`conc`

Examples

```
# generate y
x <- seq(-1, 2, by = 0.1)
n <- length(x)
y <- - x^2 + rnorm(n, .3)

# regress y on x under the shape-restriction: "smooth and concave"
ans <- cgam(y ~ s.conc(x))
knots <- ans$knots[[1]]

# make a plot
plot(x, y)
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth and concave fit", col = 2, lty = 1)
legend(1.6, 1.8, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.conc.conc` *Specify a Doubly-Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that a surface is concave in two predictors in a formula argument to cgam.

Usage

```
s.conc.conc(x1, x2, numknots = c(0, 0), knots = list(k1 = 0, k2 = 0), space = c("E", "E"))
```

Arguments

`x1`	A numeric predictor which has the same length as the response vector.
`x2`	A numeric predictor which has the same length as the response vector.
`numknots`	A vector of the number of knots used to constrain x_1 and x_2. It will not be used if the user specifies the *knots* argument and each predictor is within the range of its knots. The default is numknots = c(0, 0).
`knots`	A list of two vectors of knots used to constrain x_1 and x_2. User-defined knots will be used if each predictor is within the range of its knots. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = list(k1 = 0, k2 = 0).
`space`	A vector of the character specifying the method to create knots for x_1 and x_2. It will not be used if the user specifies the *knots* argument. If "E" is used, then equally spaced knots will be created; if "Q" is used, then a vector of equal quantiles will be created with duplicate elements removed. The number of knots is *numknots* when *numknots* is a positive integer > 4. Otherwise it is of the order $n^{1/3}$. The default is space = c("E", "E").

Details

"s.conc.conc" returns the vectors "x1" and "x2", and imposes on each vector six attributes: name, shape, numknots, knots, space and cvs.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.conc.conc"; the shape attribute is "tri_cvs"(doubly-concave); the cvs values for both vectors are FALSE. According to the value of the vector itself and its shape, numknots, knots, space and cvs attributes, the cone edges will be made by triangle spline basis functions in Meyer (2017). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.conc.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called trispl.fit

See references cited in this section for more details.

Value

The vectors x_1 and x_2. Each of them has six attributes, i.e., name: names of x_1 and x_2; shape: "tri_cvs"(doubly-concave); numknots: the numknots argument in "s.conc.conc"; knots: the knots argument in "s.conc.conc"; space: the space argument in "s.conc.conc"; cvs: two logical values indicating the monotonicity of the isotonically-constrained surface with respect to x_1 and x_2, which are both FALSE.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Estimation and inference for regression surfaces using shape-constrained splines.

See Also

`s.conv.conv`, `cgam`

Examples

```
# generate data
n <- 200
set.seed(123)
x1 <- runif(n); x2 <- runif(n)
y <- -(x1 - 1)^2 - (x2 - 3)^2 + rnorm(n)

    # regress y on x1 and x2 under the shape-restriction: "doubly-concave"
    ans <- cgam(y ~ s.conc.conc(x1, x2), nsim = 0)
    # make a 3D plot of the constrained surface
    plotpersp(ans)
```

`s.conv` *Specify a Smooth and Convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth and convex in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.conv(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s.conv" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.conv"; the shape attribute is 11("smooth and convex"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 11("smooth and convex"); numknots: the numknots argument in "s.conv"; knots: the knots argument in "s.conv"; space: the space argument in "s.conv".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`conv`

Examples

```
# generate y
x <- seq(-1, 2, by = 0.1)
n <- length(x)
y <- x^2 + rnorm(n, .3)

# regress y on x under the shape-restriction: "smooth and convex"
ans <- cgam(y ~ s.conv(x))
knots <- ans$knots[[1]]

# make a plot
plot(x, y)
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth and convex fit", col = 2, lty = 1)
legend(1.6, -1, bty = "o", "knots", pch = "X")
```

```
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.conv.conv` *Specify a Doubly-convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that a surface is convex in two predictors in a formula argument to cgam.

Usage

```
s.conv.conv(x1, x2, numknots = c(0, 0), knots = list(k1 = 0, k2 = 0), space = c("E", "E"))
```

Arguments

`x1`	A numeric predictor which has the same length as the response vector.
`x2`	A numeric predictor which has the same length as the response vector.
`numknots`	A vector of the number of knots used to constrain x_1 and x_2. It will not be used if the user specifies the *knots* argument and each predictor is within the range of its knots. The default is numknots = c(0, 0).
`knots`	A list of two vectors of knots used to constrain x_1 and x_2. User-defined knots will be used if each predictor is within the range of its knots. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = list(k1 = 0, k2 = 0).
`space`	A vector of the character specifying the method to create knots for x_1 and x_2. It will not be used if the user specifies the *knots* argument. If "E" is used, then equally spaced knots will be created; if "Q" is used, then a vector of equal quantiles will be created with duplicate elements removed. The number of knots is *numknots* when *numknots* is a positive integer > 4. Otherwise it is of the order $n^{1/3}$. The default is space = c("E", "E").

Details

"s.conv.conv" returns the vectors "x1" and "x2", and imposes on each vector six attributes: name, shape, numknots, knots, space and cvs.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.conv.conv"; the shape attribute is "tri_cvs"(doubly-convex); the cvs values for both vectors are TRUE. According to the value of the vector itself and its shape, numknots, knots, space and cvs attributes, the cone edges will be made by triangle spline basis functions in Meyer (2017). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.conv.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called trispl.fit

See references cited in this section for more details.

Value

The vectors x_1 and x_2. Each of them has six attributes, i.e., name: names of x_1 and x_2; shape: "tri_cvs"(doubly-convex); numknots: the numknots argument in "s.conv.conv"; knots: the knots argument in "s.conv.conv"; space: the space argument in "s.conv.conv"; cvs: two logical values indicating the monotonicity of the isotonically-constrained surface with respect to x_1 and x_2, which are both TRUE.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Estimation and inference for regression surfaces using shape-constrained splines.

See Also

`s.conv.conv`, `cgam`

Examples

```
# generate data
n <- 200
set.seed(123)
x1 <- runif(n); x2 <- runif(n)
y <- (x1 - 1)^2 + (x2 - 3)^2 + rnorm(n)

    # regress y on x1 and x2 under the shape-restriction: "doubly-convex"
    ans <- cgam(y ~ s.conv.conv(x1, x2), nsim = 0)
    # make a 3D plot of the constrained surface
    plotpersp(ans)
```

`s.decr` *Specify a Smooth and Decreasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth and decreasing in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.decr(x, numknots = 0, knots = 0, var.knots = 0, space = "Q", db.exp = FALSE)
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`var.knots`	The knots used in variance function estimation. User-defined knots will be used when given. The default is var.knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".
`db.exp`	The parameter will be used in variance function estimation. If db.exp = TRUE, then the errors are assumed to follow a normal distribution; otherwise, the errors are assumed to follow a double-exponential distribution. The default is db.exp = FALSE.

Details

"s.decr" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots, space, var.knots and db.exp.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.decr"; the shape attribute is 10("smooth and decreasing"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by I-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.decr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

var.knots and db.exp will be used for monotonic variance function estimation.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e, name: the name of x; shape: 10("smooth and decreasing"); numknots: the numknots argument in "s.decr"; knots: the knots argument in "s.decr"; space: the space argument in "s.decr".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`decr`

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- cubic$y

# regress y on x under the shape-restriction: "smooth and decreasing"
ans <- cgam(y ~ s.decr(x))
knots <- ans$knots[[1]]

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth and decreasing fit", col = 2, lty = 1)
legend(-.3, 8, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.decr.conc` *Specify a Smooth, Decreasing and Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth, decreasing and concave in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.decr.conc(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s.decr.conc" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.decr.conc"; the shape attribute is 16("smooth, decreasing and concave"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.decr.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 16("smooth, decreasing and concave"); numknots: the numknots argument in "s.decr.conc"; knots: the knots argument in "s.decr.conc"; space: the space argument in "s.decr.conc".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`decr.conv`, `decr`

Examples

```
data(cubic)

# extract x
x <-  cubic$x

# extract y
y <- - cubic$y

# regress y on x under the shape-restriction: "smooth, decreasing and concave"
ans <- cgam(y ~ s.decr.conc(x))
knots <- ans$knots[[1]]

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth, decreasing and concave fit", col = 2, lty = 1)
legend(1.7, 4, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.decr.conv` *Specify a Smooth, Decreasing and Convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth, decreasing and convex in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.decr.conv(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s.decr.conv" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.decr.conv"; the shape attribute is 15("smooth, decreasing and convex"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.decr.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 15("smooth, decreasing and convex"); numknots: the numknots argument in "s.decr.conv"; knots: the knots argument in "s.decr.conv"; space: the space argument in "s.decr.conv".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`decr.conv`

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- cubic$y

# regress y on x under the shape-restriction: "smooth, decreasing and convex"
ans <- cgam(y ~ s.decr.conv(x))
knots <- ans$knots[[1]]

# make a plot
par(mar = c(4, 4, 1, 1))
```

```
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth, decreasing and convex fit", col = 2, lty = 1)
legend(-.3, 9.2, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.decr.decr` *Specify a Doubly-Decreasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that a surface is decreasing in two predictors in a formula argument to cgam.

Usage

```
s.decr.decr(x1, x2, numknots = c(0, 0), knots = list(k1 = 0, k2 = 0), space = c("E", "E"))
```

Arguments

`x1`	A numeric predictor which has the same length as the response vector.
`x2`	A numeric predictor which has the same length as the response vector.
`numknots`	A vector of the number of knots used to constrain x_1 and x_2. It will not be used if the user specifies the *knots* argument and each predictor is within the range of its knots. The default is numknots = c(0, 0).
`knots`	A list of two vectors of knots used to constrain x_1 and x_2. User-defined knots will be used if each predictor is within the range of its knots. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = list(k1 = 0, k2 = 0).
`space`	A vector of the character specifying the method to create knots for x_1 and x_2. It will not be used if the user specifies the *knots* argument. If "E" is used, then equally spaced knots will be created; if "Q" is used, then a vector of equal quantiles will be created with duplicate elements removed. The number of knots is *numknots* when *numknots* is a positive integer > 4. Otherwise it is of the order $n^{1/6}$. The default is space = c("E", "E").

Details

"s.decr.decr" returns the vectors "x1" and "x2", and imposes on each vector six attributes: name, shape, numknots, knots, space and decreasing.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.decr.decr"; the shape attribute is "wps_dd"(doubly-decreasing); the decreasing values for both vectors are TRUE. According to the value of the vector itself and its shape, numknots, knots, space and decreasing attributes, the cone edges will be made by warped-plane spline basis functions in Meyer (2016). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.decr.decr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta_wps.

See references cited in this section for more details.

Value

The vectors x_1 and x_2. Each of them has six attributes, i.e., name: names of x_1 and x_2; shape: "wps_dd"(doubly-decreasing); numknots: the numknots argument in "s.decr.decr"; knots: the knots argument in "s.decr.decr"; space: the space argument in "s.decr.decr"; decreasing: two logical values indicating the monotonicity of the isotonically-constrained surface with respect to x_1 and x_2, which are both TRUE.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Estimation and inference for regression surfaces using shape-constrained splines.

See Also

`s.incr.incr`, `s.decr.incr`, `s.incr.decr`, `cgam`

Examples

```
## Not run:
  # generate data
  n <- 100
  set.seed(123)
  x1 <- runif(n)
  x2 <- runif(n)
  y <- -4 * (x1 + x2 - x1 * x2) + rnorm(n, sd = .2)

  # regress y on x1 and x2 under the shape-restriction: "doubly-decreasing"
  # using the penalized estimator
  ans <- cgam(y ~ s.decr.decr(x1, x2), pnt = TRUE)

  # make a 3D plot of the constrained surface
  plotpersp(ans)

## End(Not run)
```

`s.decr.incr` *Specify a Decreasing-Increasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that a surface is decreasing in one predictor and increasing in another in a formula argument to cgam.

Usage

```
s.decr.incr(x1, x2, numknots = c(0, 0), knots = list(k1 = 0, k2 = 0), space = c("E", "E"))
```

Arguments

`x1`	A numeric predictor which has the same length as the response vector.
`x2`	A numeric predictor which has the same length as the response vector.
`numknots`	A vector of the number of knots used to constrain x_1 and x_2. It will not be used if the user specifies the *knots* argument and each predictor is within the range of its knots. The default is numknots = c(0, 0).
`knots`	A list of two vectors of knots used to constrain x_1 and x_2. User-defined knots will be used if each predictor is within the range of its knots. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = list(k1 = 0, k2 = 0).
`space`	A vector of the character specifying the method to create knots for x_1 and x_2. It will not be used if the user specifies the *knots* argument. If "E" is used, then equally spaced knots will be created; if "Q" is used, then a vector of equal quantiles will be created with duplicate elements removed. The number of knots is *numknots* when *numknots* is a positive integer > 4. Otherwise it is of the order $n^{1/6}$. The default is space = c("E", "E").

Details

"s.decr.incr" returns the vectors "x1" and "x2", and imposes on each vector six attributes: name, shape, numknots, knots, space and decreasing.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.decr.incr"; the shape attribute is "wps_di"(decreasing-increasing); the decreasing values for "x1" and "x2" are TRUE and FALSE. According to the value of the vector itself and its shape, numknots, knots, space and decreasing attributes, the cone edges will be made by warped-plane spline basis functions in Meyer (2016). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.decr.incr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta_wps.

See references cited in this section for more details.

Value

The vectors x_1 and x_2. Each of them has six attributes, i.e., name: names of x_1 and x_2; shape: "wps_di"(decreasing-increasing); numknots: the numknots argument in "s.decr.incr"; knots: the knots argument in "s.decr.incr"; space: the space argument in "s.decr.incr"; decreasing: two logical values indicating the monotonicity of the isotonically-constrained surface with respect to x_1 and x_2, which are TRUE and FALSE.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Estimation and inference for regression surfaces using shape-constrained splines.

See Also

`s.incr.incr`, `s.incr.decr`, `s.decr.decr`, `cgam`

Examples

```
## Not run:
  # generate data
  n <- 100
  set.seed(123)
  x1 <- runif(n)
  x2 <- runif(n)
  y <- 4 * (x2 - x1) - x1 * x2 + rnorm(n, sd = .2)

  # regress y on x1 and x2 under the shape-restriction: "decreasing-increasing"
  # using the penalized estimator
  ans <- cgam(y ~ s.decr.incr(x1, x2), pnt = TRUE)

  # make a 3D plot of the constrained surface
  plotpersp(ans)

## End(Not run)
```

`s.incr` *Specify a Smooth and Increasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth and increasing in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.incr(x, numknots = 0, knots = 0, var.knots = 0, space = "Q", db.exp = FALSE)
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`var.knots`	The knots used in variance function estimation. User-defined knots will be used when given. The default is var.knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".
`db.exp`	The parameter will be used in variance function estimation. If db.exp = TRUE, then the errors are assumed to follow a normal distribution; otherwise, the errors are assumed to follow a double-exponential distribution. The default is db.exp = FALSE.

Details

"s.incr" returns the vector "x" and imposes on it seven attributes: name, shape, numknots, knots, space, var.knots and db.exp.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.incr"; the shape attribute is 9("smooth and increasing"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by I-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.incr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

var.knots and db.exp will be used for monotonic variance function estimation.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 9("smooth and increasing"); numknots: the numknots argument in "s.incr"; knots: the knots argument in "s.incr"; space: the space argument in "s.incr".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`incr`

Examples

```
data(cubic)

# extract x
x <- cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "smooth and increasing"
ans <- cgam(y ~ s.incr(x))
knots <- ans$knots[[1]]

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth and increasing fit", col = 2, lty = 1)
legend(1.7, 9.2, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.incr.conc` *Specify a Smooth, Increasing and Concave Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth, increasing and concave in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.incr.conc(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s.incr.conc" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.incr.conc"; the shape attribute is 14("smooth, increasing and concave"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.incr.conc" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 14("smooth, increasing and concave"); numknots: the numknots argument in "s.incr.conc"; knots: the knots argument in "s.incr.conc"; space: the space argument in "s.incr.conc".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics **42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics **2(3)***, 1013–1033.

See Also

`incr.conv`

Examples

```
data(cubic)

# extract x
x <- - cubic$x

# extract y
y <- - cubic$y

# regress y on x with the shape restriction: "smooth, increasing and concave"
ans <- cgam(y ~ s.incr.conc(x))
knots <- ans$knots[[1]]

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth, increasing and concave fit", col = 2, lty = 1)
legend(-.3, 4, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.incr.conv` *Specify an Smooth, Increasing and Convex Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η is smooth, increasing and convex in a predictor in a formula argument to cgam. This is the smooth version.

Usage

```
s.incr.conv(x, numknots = 0, knots = 0, space = "Q")
```

Arguments

`x`	A numeric predictor which has the same length as the response vector.
`numknots`	The number of knots used to constrain x. It will not be used if the user specifies the *knots* argument. The default is numknots = 0.
`knots`	The knots used to constrain x. User-defined knots will be used when given. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = 0.
`space`	A character specifying the method to create knots. It will not be used if the user specifies the *knots* argument. If space == "E", then equally spaced knots will be created; if space == "Q", then a vector of equal x quantiles will be created based on x with duplicate elements removed. The number of knots is *numknots* when *numknots* > 0. Otherwise it is of the order $n^{1/7}$. The default is space = "Q".

Details

"s.incr.conv" returns the vector "x" and imposes on it five attributes: name, shape, numknots, knots and space.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.incr.conv"; the shape attribute is 13("smooth, increasing and convex"). According to the value of the vector itself and its shape, numknots, knots and space attributes, the cone edges will be made by C-spline basis functions in Meyer (2008). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.incr.conv" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta in cgam.

See references cited in this section for more details.

Value

The vector x with five attributes, i.e., name: the name of x; shape: 13("smooth, increasing and convex"); numknots: the numknots argument in "s.incr.conv"; knots: the knots argument in "s.incr.conv"; space: the space argument in "s.incr.conv".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

Meyer, M. C. (2008) Inference using shape-restricted regression splines. *Annals of Applied Statistics* ***2(3)***, 1013–1033.

See Also

`incr.conv`

Examples

```
data(cubic)

# extract x
x <- cubic$x

# extract y
y <- cubic$y

# regress y on x with the shape restriction: "smooth, increasing and convex"
ans <- cgam(y ~ s.incr.conv(x))
knots <- ans$knots[[1]]

# make a plot
par(mar = c(4, 4, 1, 1))
```

```
plot(x, y, cex = .7, xlab = "x", ylab = "y")
lines(x, ans$muhat, col = 2)
legend("topleft", bty = "n", "smooth, increasing and convex fit", col = 2, lty = 1)
legend(1.7, 9.2, bty = "o", "knots", pch = "X")
points(knots, 1:length(knots)*0+min(y), pch = "X")
```

`s.incr.decr` *Specify an Increasing-Decreasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that a surface is decreasing in one predictor and increasing in another in a formula argument to cgam.

Usage

```
s.incr.decr(x1, x2, numknots = c(0, 0), knots = list(k1 = 0, k2 = 0), space = c("E", "E"))
```

Arguments

`x1`	A numeric predictor which has the same length as the response vector.
`x2`	A numeric predictor which has the same length as the response vector.
`numknots`	A vector of the number of knots used to constrain x_1 and x_2. It will not be used if the user specifies the *knots* argument and each predictor is within the range of its knots. The default is numknots = c(0, 0).
`knots`	A list of two vectors of knots used to constrain x_1 and x_2. User-defined knots will be used if each predictor is within the range of its knots. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = list(k1 = 0, k2 = 0).
`space`	A vector of the character specifying the method to create knots for x_1 and x_2. It will not be used if the user specifies the *knots* argument. If "E" is used, then equally spaced knots will be created; if "Q" is used, then a vector of equal quantiles will be created with duplicate elements removed. The number of knots is *numknots* when *numknots* is a positive integer > 4. Otherwise it is of the order $n^{1/6}$. The default is space = c("E", "E").

Details

"s.incr.decr" returns the vectors "x1" and "x2", and imposes on each vector six attributes: name, shape, numknots, knots, space and decreasing.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.incr.decr"; the shape attribute is "wps_id"(increasing-decreasing); the decreasing values for "x1" and "x2" are TRUE and FALSE. According to the value of the vector itself and its shape, numknots, knots, space and decreasing attributes, the cone edges will be made by warped-plane spline basis functions in Meyer (2016). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.incr.decr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta_wps.

See references cited in this section for more details.

Value

The vectors x_1 and x_2. Each of them has six attributes, i.e., name: names of x_1 and x_2; shape: "wps_id"(increasing-decreasing); numknots: the numknots argument in "s.incr.decr"; knots: the knots argument in "s.incr.decr"; space: the space argument in "s.incr.decr"; decreasing: two logical values indicating the monotonicity of the isotonically-constrained surface with respect to x_1 and x_2, which are FALSE and TRUE.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Estimation and inference for regression surfaces using shape-constrained splines.

See Also

`s.incr.incr`, `s.decr.decr`, `s.decr.incr`, `cgam`

Examples

```
## Not run:
  # generate data
  n <- 100
  set.seed(123)
  x1 <- runif(n)
  x2 <- runif(n)
  y <- 4 * (x1 - x2) - x1 * x2 + rnorm(n, sd = .2)

  # regress y on x1 and x2 under the shape-restriction: "increasing-decreasing"
  # using the penalized estimator
  ans <- cgam(y ~ s.incr.decr(x1, x2), pnt = TRUE)

  # make a 3D plot of the constrained surface
  plotpersp(ans)

## End(Not run)
```

`s.incr.incr` *Specify a Doubly-Increasing Shape-Restriction in a CGAM Formula*

Description

A symbolic routine to define that a surface is increasing in two predictors in a formula argument to cgam.

Usage

```
s.incr.incr(x1, x2, numknots = c(0, 0), knots = list(k1 = 0, k2 = 0), space = c("E", "E"))
```

Arguments

`x1`	A numeric predictor which has the same length as the response vector.
`x2`	A numeric predictor which has the same length as the response vector.
`numknots`	A vector of the number of knots used to constrain x_1 and x_2. It will not be used if the user specifies the *knots* argument and each predictor is within the range of its knots. The default is numknots = c(0, 0).
`knots`	A list of two vectors of knots used to constrain x_1 and x_2. User-defined knots will be used if each predictor is within the range of its knots. Otherwise, *numknots* and *space* will be used to create knots. The default is knots = list(k1 = 0, k2 = 0).
`space`	A vector of the character specifying the method to create knots for x_1 and x_2. It will not be used if the user specifies the *knots* argument. If "E" is used, then equally spaced knots will be created; if "Q" is used, then a vector of equal quantiles will be created with duplicate elements removed. The number of knots is *numknots* when *numknots* is a positive integer > 4. Otherwise it is of the order $n^{1/6}$. The default is space = c("E", "E").

Details

"s.incr.incr" returns the vectors "x1" and "x2", and imposes on each vector six attributes: name, shape, numknots, knots, space and decreasing.

The name attribute is used in the subroutine plotpersp; the numknots, knots and space attributes are the same as the numknots, knots and space arguments in "s.incr.incr"; the shape attribute is "wps_ii"(doubly-increasing); the decreasing values for both vectors are FALSE. According to the value of the vector itself and its shape, numknots, knots, space and decreasing attributes, the cone edges will be made by warped-plane spline basis functions in Meyer (2016). The cone edges are a set of basis employed in the hinge algorithm.

Note that "s.incr.incr" does not make the corresponding cone edges itself. It sets things up to a subroutine called makedelta_wps.

See references cited in this section for more details.

Value

The vectors x_1 and x_2. Each of them has six attributes, i.e., name: names of x_1 and x_2; shape: "wps_ii"(doubly-increasing); numknots: the numknots argument in "s.incr.incr"; knots: the knots argument in "s.incr.incr"; space: the space argument in "s.incr.incr"; decreasing: two logical values indicating the monotonicity of the isotonically-constrained surface with respect to x_1 and x_2, which are both FALSE.

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2017) Estimation and inference for regression surfaces using shape-constrained splines.

See Also

`s.decr.decr`, `s.decr.incr`, `cgam`

Examples

```
## Not run:
  # generate data
  n <- 100
  set.seed(123)
  x1 <- runif(n)
  x2 <- runif(n)
  y <- 4 * (x1 + x2 - x1 * x2) + rnorm(n, sd = .2)

  # regress y on x1 and x2 under the shape-restriction: "doubly-increasing"
  # using the penalized estimator
  ans <- cgam(y ~ s.incr.incr(x1, x2), pnt = TRUE)

  # make a 3D plot of the constrained surface
  plotpersp(ans)

## End(Not run)
```

`shapeselect` *Shape selection for additive components with shape or order constraints*

Description

Fits an additive model by selecting variables with shape or order constraints for each predictor using Cone Information Bayesian Criterion (CBIC). The function can perform forward selection or stepwise (forward/backward) updates over a set of candidate shapes appropriate to each predictor type.

Usage

```
shapeselect(
  xmat,
  y,
  types = NULL,
  direction = c("both", "forward"),
  type = "stepwise",
  infocrit = c("CBIC", "CAIC"),
  family = gaussian,
  control = list(),
  ...
)
```

Arguments

`xmat`	A data frame of predictors. Columns may be numeric or factors. Column names are used in output when available. If x is numeric, then the shape options are increasing, decreasing, convex, concave or combination of monotonicity and convexity; if x is a factor, then it can have monotonic or convex ordering options; if x is a factor, the option can also be "in" or "out-of" the model, or "in", "out-of" the model, or following a "tree" ordering, depending on the `types` parameter.
`y`	A response vector of length `nrow(xmat)`. It can be either numeric or binary.
`types`	Optional character vector of length `ncol(xmat)` specifying the predictor type for each column of `xmat`. This is used to determine which shape set is eligible for each predictor. Values used for this parameter include `"num"`/`"numeric"` (continuous), `"ord"`/`"ordinal"` (ordinal/categorical), `"tree"` (categorical), and `"nom"`/`"nominal"` (categorical). Default is `NULL`, but the user must provide a value for each column of xmat; otherwise the function will throw an error message.
`direction`	Direction of the search. `"forward"` performs forward selection only. `"both"` allows backward revisions (stepwise) after additions.
`type`	Search strategy. `"stepwise"` adds predictors while allowing backward revisions. `"classo"` then applies shrinkage/selection (e.g., via cross-validated lasso). The available option is `"stepwise"` for now.
`infocrit`	Information criterion used to decide adding or removing a predictor, and shape updates. Currently supports `"CBIC"` and `"CAIC"`.
`family`	A GLM family. May be a family function or a character string naming one. Available families are `gaussian` and `binomial`.
`control`	A list of control parameters. Any entries not supplied fall back to `shapeselect_control()`. Additional named arguments in `...` override both defaults and `control`.
`...`	Additional control parameters (passed as overrides). These are merged into `control` after defaults, so they take highest priority.

Details

The algorithm begins with an intercept-only fit, then evaluates each inactive predictor under a set of allowable shapes (depending on `types`). At each step it selects the predictor/shape combination that improves the chosen information criterion. When `direction = "both"`, it checks non-chosen shapes for already-selected predictors via backward steps.

Shape and order choices in this implementation include

1. smooth effects: monotone increasing/decreasing, convex/concave when `set4 = TRUE`, and combinations `set4 = FALSE`, if the corresponding value in `types` is `"num"/"numeric"` (x is continuous).
2. non-smooth effects: monotone increasing/decreasing and convex/concave (when `set4 = TRUE`), or their combinations (when `set4 = FALSE`), if the corresponding value in `types` is `"ord"/"ordinal"` (i.e., x is ordinal, categorical, or numeric but treated as discrete with an inherent ordering). in `types` is `"ord"/"ordinal"` (x is ordinal/categorical/having numeric values but treated as discrete values with ordering).
3. in or out-of the model, if the corresponding value in `types` is `"nom"/"nominal"` (x is categorical)
4. in, out-of the model or following a tree ordering, if the corresponding value in `types` is `"tree"` (i.e., x is categorical or numeric but treated as discrete with a tree ordering).

Value

An object of class `c("shapeselect","cgam")`, a list containing (among others):

mod_tried A matrix recording the selection path (chosen shapes, criterion values, adding or removing a predictor).

shps Numeric shape codes for each predictor (0 indicates excluded/flat).

shapes Character labels for the selected shapes.

etahat Fitted linear predictor values.

muhat Fitted mean response on the data scale (`family$linkinv(etahat)`).

etacomps Matrix of fitted component contributions by predictor (rows).

cic_vec Criterion values along the path.

cvec Shrinkage/selection vector (used when `type = "classo"`).

ddkeep_x_lst, ddkeep_z_lst Stored design/basis components for numeric and categorical predictors used for prediction and plotting.

Examples

```
library(MASS)
data("Rubber", package = "MASS")
y <- Rubber[, 1]
xmat <- Rubber[, -1]
ans <- shapeselect(xmat, y, types = c("num", "num"), pnt = FALSE)

print (ans$shapes)

plot(ans)
```

`shapeselect_control` *Control parameters for* `shapeselect()`

Description

Creates a list of parameters used by `shapeselect()` to control basis construction, information-criterion calibration, simulation settings, parallel computing etc.

Usage

```
shapeselect_control(
  knots_lst = NULL,
  boundary.knots = NULL,
  space = "E",
  nsim = 200,
  nfolds = 5,
  eps = 1e-08,
  pnt = TRUE,
  multicore = getOption("cgam.parallel"),
  a = NULL,
  maxedf = FALSE,
  add_edfu = 0,
  add_kts = NULL,
  shrinketa = FALSE,
  catnrep = FALSE,
  standardize = TRUE,
  center = TRUE,
  verbose = FALSE,
  set4 = TRUE
)
```

Arguments

`knots_lst`	Optional list specifying knots for spline basis functions, one element per predictor. Use `NULL` to let `shapeselect()` (and its helper routines) choose defaults.
`boundary.knots`	Optional list specifiying knots for numeric predictors.
`space`	Character string indicating the knots being equally spaced for spline bases in underlying fitting routines (e.g., `"E"`).
`nsim`	Integer; number of simulations used when estimating mean degrees of freedom terms used in variable and shape selection criteria CBIC.
`nfolds`	Integer; number of folds used for cross-validation when `type = "classo"` (shrinkage selection). It is not in use for now.
`eps`	Numeric tolerance used for detecting near-zero signal.
`pnt`	Logical; whether to use penalized spline estimator.

`multicore`	Logical/integer; parallel setting passed to `shapeselect()` via the `parallel` control entry. Defaults to `getOption("cgam.parallel")`.
`a`	Optional numeric; calibration constant used in the modified CBIC penalty term. If `NULL`, `shapeselect()` chooses a value based on `n`, `p`, and the implied total number of basis functions.
`maxedf`	Logical; if `TRUE`, enforce an effective degrees of freedom cap in internal forward-step routines (as implemented).
`add_edfu`	Numeric; optional additive adjustment to the unconstrained EDF used in the information criterion calculations.
`add_kts`	Integer; optional adjustment to the number of knots used in spline bases. If `NULL`, `shapeselect()` chooses a default depending on `pnt`.
`shrinketa`	Logical; if `TRUE` and `type = "classo"`, shrink fitted component contributions by the selected shrinkage parameters. It is not in use for now.
`catnrep`	Logical; if `TRUE`, print the iteration counter during the selection path.
`standardize`	Logical; if `TRUE`, standardize numeric predictors by their standard deviations.
`center`	Logical; if `TRUE`, center numeric predictors when scaling them.
`verbose`	Logical; if `TRUE`, print the model-path tracking matrix as the algorithm runs.
`set4`	Logical; optional flag passed through to internal routines used to restrict or modify the candidate shape set to be increasing, decreasing, convex and concave; if false, then combinations of monotonicity and convexity will be added

Value

A named list suitable for the `control` argument of `shapeselect()`.

Examples

```
ctrl <- shapeselect_control(nsim = 100, nfolds = 10, verbose = TRUE)
ctrl
```

`testpar` *Parametric Monotone/Quadratic vs Smooth Constrained Test*

Description

Performs a hypothesis test comparing a parametric model (e.g., linear, quadratic, warped plane) to a constrained smooth alternative under shape restrictions.

Usage

```
testpar(
  formula0,
  formula,
  data,
  family = gaussian(link = "identity"),
  ps = NULL,
  edfu = NULL,
  nsim = 1000,
  multicore = getOption("cgam.parallel"),
  method = "testpar.fit",
  arp = FALSE,
  p = NULL,
  space = "E",
  ...
)
```

Arguments

`formula0`	A model formula representing the null hypothesis (e.g., linear or quadratic).
`formula`	A model formula under the alternative hypothesis, possibly with shape constraints.
`data`	A data frame or environment containing the variables in the model.
`family`	A description of the error distribution and link function to be used in the model (e.g., `gaussian()`). Gaussian and binomial are now included.
`ps`	User-defined penalty term. If `NULL`, optimal values are estimated.
`edfu`	User-defined unconstrained effective degrees of freedom.
`nsim`	Number of simulations to perform to get the mixture distribution of the test statistic. (default is 1000).
`multicore`	Logical. Whether to enable parallel computation (default uses global option `cgam.parallel`).
`method`	A character string (default is `"testpar.fit"`), currently unused.
`arp`	Logical. If `TRUE`, uses autoregressive structure estimation.
`p`	Integer vector or value specifying AR(p) order. Ignored if `arp = FALSE`.
`space`	Character. Either `"Q"` for quantile-based knots or `"E"` for evenly spaced knots. Default is `"E"`.
`...`	Additional arguments passed to internal functions.

Value

A list containing test statistics, fitted values, coefficients, and other relevant objects.

Examples

```
# Example 1: Linear vs. monotone alternative
set.seed(123)
n <- 100
x <- sort(runif(n))
y <- 2 * x + rnorm(n)
dat <- data.frame(y = y, x = x)
ans <- testpar(formula0 = y ~ x, formula = y ~ s.incr(x), multicore = FALSE, nsim = 10)
ans$pval
summary(ans)
## Not run:
# Example 2: Binary response: linear vs. monotone alternative
set.seed(123)
n <- 100
x <- runif(n)
eta <- 8 * x - 4
mu <- exp(eta) / (1 + exp(eta))
y <- rbinom(n, size = 1, prob = mu)

ans <- testpar(formula0 = y ~ x, formula = y ~ s.incr(x),
family = binomial(link = "logit"), nsim = 10)
summary(ans)

xs = sort(x)
ord = order(x)
 par(mfrow = c(1,1))
 plot(x, y, cex = .2)
 lines(xs, binomial(link="logit")$linkinv(ans$etahat0)[ord], col=4, lty=4)
 lines(xs, binomial(link="logit")$linkinv(ans$etahat)[ord], col=2, lty=2)
 lines(xs, mu[ord])
 legend("topleft", legend = c("H0 fit (etahat0)", "H1 fit (etahat)", "True mean"),
 col = c(4, 2, 1), lty = c(4, 2, 1), bty = "n", cex = 0.8)

## End(Not run)

# Example 3: Bivariate warped-plane surface test
set.seed(1)
n <- 100
x1 <- runif(n)
x2 <- runif(n)
mu <- 4 * (x1 + x2 - x1 * x2)
eps <- rnorm(n)
y <- mu + eps

# options(cgam.parallel = TRUE) #allow parallel computation
ans <- testpar(formula0 = y ~ x1 * x2, formula = y ~ s.incr.incr(x1, x2), nsim = 10)
summary(ans)

par(mfrow = c(1, 2))
persp(ans$k1, ans$k2, ans$etahat0_surf, th=-50, main = 'H0 fit')
persp(ans$k1, ans$k2, ans$etahat_surf, th=-50, main = 'H1 fit')
```

```
## Not run:
# Example 4: AR(1) error, quadratic vs smooth convex
set.seed(123)
n = 100
x = runif(n)
mu = 1+x+2*x^2
eps = arima.sim(n = n, list(ar = c(.4)), sd = 1)
y = mu + eps
ans = testpar(y~x^2, y~s.conv(x), arp=TRUE, p=1, nsim=10)
ans$pval
ans$phi #autoregressive coefficient estimate

## End(Not run)
## Not run:
# Example 5: Three additive components with shape constraints
n <- 100
x1 <- runif(n)
x2 <- runif(n)
x3 <- runif(n)
z <- rnorm(n)
mu1 <- 3 * x1 + 1
mu2 <- x2 + 3 * x2^2
mu3 <- -x3^2
mu <- mu1 + mu2 + mu3
y <- mu + rnorm(n)

ans <- testpar(formula0 = y ~ x1 + x2^2 + x3^2 + z,
formula = y ~ s.incr(x1) + s.incr.conv(x2) + s.decr.conc(x3)
 + z, family = "gaussian", nsim = 10)
ans$pval
summary(ans)

## End(Not run)
```

`tree` *Specify a Tree-Ordering in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η has a tree-ordering in a predictor in a formula argument to cgam.

Usage

```
tree(x, pl = NULL)
```

Arguments

`x`	A numeric vector which has the same length as the response vector. Note that the placebo level of x must be 0.
`pl`	The placebo level.

Details

"tree" returns the vector "x" and imposes on it two attributes: name and shape.

The name attribute is used in the subroutine plotpersp; the shape attribute is "tree", and according to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains that η has a tree-ordering in "x" will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "tree" does not make the corresponding cone edges itself. It sets things up to a sub-routine called tree.fun in cgam which will make the cone edges. A tree-ordering is a partial ordering: For a categorical variable x, if there are treatment levels $x_1, \ldots, x_k$, where x_1 is a placebo, we compare $x_i, i = 2, \ldots, k$ with x_1, and not have any other comparable pairs.

See references cited in this section for more details.

Value

The vector x with two attributes, i.e., name: the name of x; shape: "tree".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`umbrella`

Examples

```
# generate y
set.seed(123)
n <- 100
x <- rep(0:4, each = 20)
z <- rep(c("a", "b"), 50)
y <- x + I(z == "a") + rnorm(n, 1)
xu <- unique(x)

# regress y on x under the tree-ordering restriction
fit.tree <- cgam(y ~ tree(x) + factor(z))

# make a plot
plot(x, y, cex = .7)
mua = unique(fit.tree$muhat)[unique(z) == "a"]
points(xu, unique(fit.tree$muhat)[unique(z) == "a"], pch = '+', col = 4, cex = 3)
legend(0,7.5, bty = "n", "tree-ordering fit: z = 'a'", col = 4, pch = '+', cex = 1.3)
mub = unique(fit.tree$muhat)[unique(z) == "b"]
points(xu, unique(fit.tree$muhat)[unique(z) == "b"], pch = '+', col = 2, cex = 3)
legend(0,8.5, bty = "n", "tree-ordering fit: z = 'b'", col = 2, pch = '+', cex = 1.3)
```

`umbrella` *Specify an Umbrella-Ordering in a CGAM Formula*

Description

A symbolic routine to define that the systematic component η has an umbrella-ordering in a predictor in a formula argument to cgam.

Usage

```
umbrella(x)
```

Arguments

`x`	A numeric vector which has the same length as the response vector.

Details

"umbrella" returns the vector "x" and imposes on it two attributes: name and shape.

The name attribute is used in the subroutine plotpersp; the shape attribute is "umbrella", and to the value of the vector itself and its shape attribute, the cone edges of the cone generated by the constraint matrix, which constrains that η has an umbrella-ordering in "x" will be made. The cone edges are a set of basis employed in the hinge algorithm.

Note that "umbrella" does not make the corresponding cone edges itself. It sets things up to a subroutine called umbrella.fun in cgam which will make the cone edges. An umbrella-ordering is a partial ordering: Suppose we have a x_0 that is known to be a "mode" so that for $x, y >= x_0$, we have a binary relation between x and y if $x <= y$ and for $x, y <= x_0$ we have the opposite binary relation if $x <= y$, but if $x < x_0$ and $y > x_0$, there is no such binary relation.

See references cited in this section for more details.

Value

The vector x with two attributes, i.e., name: the name of x; shape: "umbrella".

Author(s)

Mary C. Meyer and Xiyue Liao

References

Meyer, M. C. (2013b) A simple new algorithm for quadratic programming with applications in statistics. *Communications in Statistics* ***42(5)***, 1126–1139.

See Also

`tree`

Examples

```
# generate y
set.seed(123)
n <- 20
x <- seq(-2, 2, length = n)
y <- - x^2 + rnorm(n)

# regress y on x under the umbrella-ordering restriction
fit <- cgam(y ~ umbrella(x))

# make a plot
par(mar = c(4, 4, 1, 1))
plot(x, y, cex = .7, ylab = "y")
lines(x, fit$muhat, col = 2)
legend("topleft", bty = "n", "umbrella-ordering fit", col = 2, lty = 1)
```

Index