Package 'npreg'

April 23, 2021

Type Package

Title Nonparametric Regression via Smoothing Splines

Version 1.0-6

Date 2021-04-22

Author Nathaniel E. Helwig <helwig@umn.edu>

Maintainer Nathaniel E. Helwig <helwig@umn.edu>

Description Multiple and generalized nonparametric regression using smoothing spline ANOVA models and generalized additive models, as described in Helwig (2020) <doi:10.4135/9781526421036885885>. Includes support for Gaussian and non-Gaussian responses, smoothers for multiple types of predictors, interactions between smoothers of mixed types, and eight different methods for smoothing parameter selection.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-04-23 08:20:05 UTC

R topics documented:

`bin.sample` *Bin Sample a Vector, Matrix, or Data Frame*

Description

Bin elements of a vector (or rows of a matrix/data frame) and randomly sample a specified number of elements from each bin. Returns sampled data and (optionally) indices of sampled data and/or breaks for defining bins.

Usage

```
bin.sample(x, nbin = 5, size = 1, equidistant = FALSE,
           index.return = FALSE, breaks.return = FALSE)
```

Arguments

`x`	Vector, matrix, or data frame to bin sample. Factors are allowed.
`nbin`	Number of bins for each variable (defaults to 5 bins for each dimension of `x`). If `length(bins) != ncol(x)`, then `nbin[1]` is used for each variable.
`size`	Size of sample to randomly draw from each bin (defaults to 1).
`equidistant`	Should bins be defined equidistantly for each predictor? If `FALSE` (default), sample quantiles define bins for each predictor. If `length(equidistant) != ncol(x)`, then `equidistant[1]` is used for each variable.
`index.return`	If `TRUE`, returns the (row) indices of the bin sampled observations.
`breaks.return`	If `TRUE`, returns the (lower bounds of the) breaks for the binning.

Details

For a single variable, the unidimensional bins are defined using the `.bincode` function. For multiple variables, the multidimensional bins are defined using the algorithm described in the appendix of Helwig et al. (2015), which combines the unidimensional bins (calculated via `.bincode`) into a multidimensional bin code.

Value

If `index.return = FALSE` and `breaks.return = FALSE`, returns the bin sampled `x` observations.

If `index.return = TRUE` and/or `breaks.return = TRUE`, returns a list with elements:

`x`	bin sampled `x` observations.
`ix`	row indices of bin sampled observations (if `index.return = TRUE`).
`bx`	lower bounds of breaks defining bins (if `breaks.return = TRUE`).

Note

For factors, the number of bins is automatically defined to be the number of levels.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E., Gao, Y., Wang, S., & Ma, P. (2015). Analyzing spatiotemporal trends in social media data via smoothing spline analysis of variance. *Spatial Statistics, 14*(C), 491-504. doi: 10.1016/j.spasta.2015.09.002

See Also

`.bincode` for binning a numeric vector

Examples

```
##########   EXAMPLE 1   ##########
### unidimensional binning

# generate data
x <- seq(0, 1, length.out = 101)

# bin sample (default)
set.seed(1)
bin.sample(x)

# bin sample (return indices)
set.seed(1)
xs <- bin.sample(x, index.return = TRUE)
xs$x             # sampled data
x[xs$ix]         # indexing sampled data

# bin sample (return indices and breaks)
set.seed(1)
xs <- bin.sample(x, index.return = TRUE, breaks.return = TRUE)
xs$x             # sampled data
x[xs$ix]         # indexing sampled data
xs$bx            # breaks



##########   EXAMPLE 2   ##########
### bidimensional binning

# generate data
x <- expand.grid(x1 = seq(0, 1, length.out = 101),
                 x2 = seq(0, 1, length.out = 101))

# bin sample (default)
```

```
set.seed(1)
bin.sample(x)

# bin sample (return indices)
set.seed(1)
xs <- bin.sample(x, index.return = TRUE)
xs$x             # sampled data
x[xs$ix,]        # indexing sampled data

# bin sample (return indices and breaks)
set.seed(1)
xs <- bin.sample(x, index.return = TRUE, breaks.return = TRUE)
xs$x             # sampled data
x[xs$ix,]        # indexing sampled data
xs$bx            # breaks

# plot breaks and 25 bins
plot(xs$bx, xlim = c(0, 1), ylim = c(0, 1),
     xlab = "x1", ylab = "x2", main = "25 bidimensional bins")
grid()
text(xs$bx + 0.1, labels = 1:25)
```

---

gsm *Fit a Generalized Smooth Model*

Description

Fits a generalized semi- or nonparametric regression model with the smoothing parameter selected via one of seven methods: GCV, OCV, GACV, ACV, PQL, AIC, or BIC.

Usage

```
gsm(formula, family = gaussian, data, weights, types = NULL, tprk = TRUE,
    knots = NULL, update = TRUE, spar = NULL, lambda = NULL, control = list(),
    method = c("GCV", "OCV", "GACV", "ACV", "PQL", "AIC", "BIC"))
```

Arguments

formula	Object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Uses the same syntax as lm and glm.
family	Description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function, or the result of a call to a family function. See the "Family Objects" section for details.
data	Optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which sm is called.

`weights` Optional vector of weights to be used in the fitting process. If provided, weighted (penalized) likelihood estimation is used. Defaults to all 1.

`types` Named list giving the type of smooth to use for each predictor. If `NULL`, the type is inferred from the data. See "Types of Smooths" section for details.

`tprk` Logical specifying how to parameterize smooth models with multiple predictors. If `TRUE` (default), a **t**ensor **p**roduct **r**eproducing **k**ernel function is used to represent the function. If `FALSE`, a tensor product of marginal kernel functions is used to represent the function. See the "Multiple Smooths" section for details.

`knots` Spline knots for the estimation of the nonparametric effects. For models with multiple predictors, the knot specification will depend on the `tprk` input. See the "Choosing Knots" section for details

`update` If `TRUE`, steps 1-2 of Gu and Wahba's (1991) algorithm 3.2 are used to update the "extra" smoothing parameters. If `FALSE`, only step 1 of algorithm 3.2 is used, so each effect is given equal influence on the penalty. Only applicable when multiple smooth terms are included.

`spar` Smoothing parameter. Typically (but not always) in the range $(0, 1]$. If specified `lambda = 256^(3*(spar-1))`.

`lambda` Computational smoothing parameter. This value is weighted by n to form the penalty coefficient (see Details). Ignored if `spar` is provided.

`control` Optional list with named components controlling the root finding when the smoothing parameter spar is computed, i.e., missing or NULL, see below.

Note that spar is only searched for in the interval $[lower, upper]$.

lower: lower bound for spar; defaults to 0.

upper: upper bound for spar; defaults to 1.

tol: the absolute precision (**tol**erance) used by `optimize`; defaults to 1e-8.

`method` Method for selecting the smoothing parameter. Ignored if `lambda` is provided.

Details

Letting $\eta_i = \eta(x_i)$ with $x_i = (x_{i1}, \ldots, x_{ip})$, the function is represented as

$$\eta = X\beta + Z\gamma$$

where the basis functions in X span the null space (i.e., parametric effects), and Z contains the kernel function(s) of the contrast space (i.e., nonparametric effects) evaluated at all combinations of observed data points and knots. The vectors β and γ contain unknown basis function coefficients.

Let $\mu_i = E(y_i)$ denote the mean of the i-th response. The unknown function is related to the mean μ_i such as

$$g(\mu_i) = \eta_i$$

where $g()$ is a known link function. Note that this implies that $\mu_i = g^{-1}(\eta_i)$ given that the link function is assumed to be invertible.

The penalized likelihood estimation problem has the form

$$-\sum_{i=1}^{n}[y_i\theta_i - b(\theta_i)] + n\lambda\gamma' Q\gamma$$

where θ_i is the canonical parameter, $b()$ is a known function that depends on the chosen family, and Q is the penalty matrix. Note that $\theta_i = g_0(\mu_i)$ where g_0 is the canonical link function. This implies that $\theta_i = \eta_i$ when the chosen link function is canonical, i.e., when $g = g_0$.

Value

An object of class "gsm" with components:

`linear.predictors`
: the linear fit on link scale. See the Note for obtaining the fitted values on the response scale.

`se.lp`
: the standard errors of the linear predictors.

`deviance`
: up to a constant, minus twice the maximized log-likelihood. Where sensible, the constant is chosen so that a saturated model has deviance zero.

`cv.crit`
: the cross-validation criterion.

`df`
: the estimated degrees of freedom (Df) for the fit model.

`nsdf`
: the degrees of freedom (Df) for the null space.

`r.squared`
: the squared correlation between response and fitted values.

`dispersion`
: the estimated dispersion parameter.

`logLik`
: the log-likelihood.

`aic`
: Akaike's Information Criterion.

`bic`
: Bayesian Information Criterion.

`spar`
: the value of `spar` computed or given, i.e., $s = 1 + \log_{256}(\lambda)/3$

`lambda`
: the value of λ corresponding to `spar`, i.e., $\lambda = 256^{3*(s-1)}$.

`penalty`
: the smoothness penalty $\gamma' Q \gamma$.

`coefficients`
: the basis function coefficients used for the fit model.

`cov.sqrt`
: the square-root of the covariance matrix of `coefficients`. Note: `tcrossprod(cov.sqrt)` reconstructs the covariance matrix.

`specs`
: a list with information used for prediction purposes:

  knots the spline knots used for each predictor.

  thetas the "extra" tuning parameters used to weight the penalties.

  xrng the ranges of the predictor variables.

  xlev the factor levels of the predictor variables (if applicable).

  tprk logical controlling the formation of tensor product smooths.

`data`
: the data used to fit the model.

`types`
: the type of smooth used for each predictor.

`terms`
: the terms included in the fit model.

`method`
: the `method` used for smoothing parameter selection. Will be `NULL` if `lambda` was provided.

`formula`
: the formula specifying the fit model.

`call`
: the matched call.

`family`	the input family evaluated as a function using .
`iter`	the number of iterations of IRPLS used.
`residuals`	the working (IRPLS) residuals from the fitted model.
`null.deviance`	the deviance of the null model (i.e., intercept only).

Family Objects

Supported families and links include:

`family`	`link`
binomial	logit, probit, cauchit, log, cloglog
gaussian	identity, log, inverse
Gamma	inverse, identity, log
inverse.gaussian	1/mu^2, inverse, identity, log
poisson	log, identity, sqrt
NegBin	log, identity, sqrt

See `NegBin` for information about the Negative Binomial family.

Methods

The smoothing parameter can be selected using one of seven methods:
Generalized Cross-Validation (GCV)
Ordinary Cross-Validation (OCV)
Generalized Approximate Cross-Validation (GACV)
Approximate Cross-Validation (ACV)
Penalized Quasi-Likelihood (PQL)
Akaike's Information Criterion (AIC)
Bayesian Information Criterion (BIC)

Types of Smooths

The following codes specify the spline types:

par	Parametric effect (factor, integer, or numeric).
nom	Nominal smoothing spline (unordered factor).
ord	Ordinal smoothing spline (ordered factor).
lin	Linear smoothing spline (integer or numeric).
cub	Cubic smoothing spline (integer or numeric).
qui	Quintic smoothing spline (integer or numeric).
per	Periodic smoothing spline (integer or numeric).
sph	Spherical spline (matrix with $d = 3$ columns).
tps	Thin-plate spline (matrix with $d \geq 1$ columns).

For finer control of some specialized spline types:

per.lin	Linear periodic spline ($m = 1$).

per.cub	Cubic periodic spline ($m = 2$).
per.qui	Quintic periodic spline ($m = 3$).
sph.lin	Linear spherical spline ($m = 1$).
sph.cub	Cubic spherical spline ($m = 2$).
sph.qui	Quintic spherical spline ($m = 3$).
tps.lin	Linear thin-plate spline ($m = 1$).
tps.cub	Cubic thin-plate spline ($m = 2$).
tps.qui	Quintic thin-plate spline ($m = 3$).

For details on the spline kernel functions, see `basis.nom` (nominal), `basis.ord` (ordinal), `basis.poly` (polynomial), `basis.sph` (spherical), and `basis.tps` (thin-plate).

Choosing Knots

If `tprk = TRUE`, the four options for the `knots` input include:

1. a scalar giving the total number of knots to sample
2. a vector of integers indexing which rows of data are the knots
3. a list with named elements giving the marginal knot values for each predictor (to be combined via `expand.grid`)
4. a list with named elements giving the knot values for each predictor (requires the same number of knots for each predicto

If `tprk = FALSE`, the three options for the `knots` input include:

1. a scalar giving the common number of knots for each continuous predictor
2. a list with named elements giving the number of marginal knots for each predictor
3. a list with named elements giving the marginal knot values for each predictor

Multiple Smooths

Suppose `formula = y ~ x1 + x2` so that the model contains additive effects of two predictor variables.

The k-th predictor's marginal effect can be denoted as

$$f_k = X_k\beta_k + Z_k\gamma_k$$

where X_k is the n by m_k null space basis function matrix, and Z_k is the n by r_k contrast space basis function matrix.

If `tprk = TRUE`, the null space basis function matrix has the form $X = [1, X_1, X_2]$ and the contrast space basis function matrix has the form

$$Z = \theta_1 Z_1 + \theta_2 Z_2$$

where the θ_k are the "extra" smoothing parameters. Note that Z is of dimension n by $r = r_1 = r_2$.

If `tprk = FALSE`, the null space basis function matrix has the form $X = [1, X_1, X_2]$, and the contrast space basis function matrix has the form

$$Z = [\theta_1 Z_1, \theta_2 Z_2]$$

where the θ_k are the "extra" smoothing parameters. Note that Z is of dimension n by $r = r_1 + r_2$.

Note

The fitted values on the response scale can be obtained using
`ginv <-object$family$linkinv`
`fit <-ginv(object$linear.predictors)`
where `object` is the fit "gsm" object.

For models with multiple predictors, the `predict.gsm` function may be more useful.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik, 31*, 377-403. doi: 10.1007/BF01404567

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Gu, C. and Wahba, G. (1991). Minimizing GCV/GML scores with multiple smoothing parameters via the Newton method. *SIAM Journal on Scientific and Statistical Computing, 12(2)*, 383-398. doi: 10.1137/0912021

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), SAGE Research Methods Foundations. doi: 10.4135/9781526421036885885

Helwig, N. E. (2020+). Spectrally sparse nonparametric regression via elastic net regularized smoothers. *Journal of Computational and Graphical Statistics*. doi: 10.1080/10618600.2020.1806855

See Also

`summary.gsm` for summarizing `gsm` objects.

`predict.gsm` for predicting from `gsm` objects.

`sm` for fitting smooth models to Gaussian data.

Examples

```
##########   EXAMPLE 1   ##########
### 1 continuous predictor

# generate data
n <- 1000
x <- seq(0, 1, length.out = n)
fx <- 3 * x + sin(2 * pi * x) - 1.5

# gaussian (default)
set.seed(1)
y <- fx + rnorm(n, sd = 1/sqrt(2))
mod <- gsm(y ~ x, knots = 10)
mean((mod$linear.predictors - fx)^2)
```

```
# compare to result from sm (they are identical)
mod.sm <- sm(y ~ x, knots = 10)
mean((mod$linear.predictors - mod.sm$fitted.values)^2)

# binomial (no weights)
set.seed(1)
y <- rbinom(n = n, size = 1, p = 1 / (1 + exp(-fx)))
mod <- gsm(y ~ x, family = binomial, knots = 10)
mean((mod$linear.predictors - fx)^2)

# binomial (w/ weights)
set.seed(1)
w <- as.integer(rep(c(10,20,30,40,50), length.out = n))
y <- rbinom(n = n, size = w, p = 1 / (1 + exp(-fx))) / w
mod <- gsm(y ~ x, family = binomial, weights = w, knots = 10)
mean((mod$linear.predictors - fx)^2)

# poisson
set.seed(1)
y <- rpois(n = n, lambda = exp(fx))
mod <- gsm(y ~ x, family = poisson, knots = 10)
mean((mod$linear.predictors - fx)^2)

# negative binomial (known theta)
set.seed(1)
y <- rnbinom(n = n, size = 1/2, mu = exp(fx))
mod <- gsm(y ~ x, family = NegBin(theta = 1/2), knots = 10)
mean((mod$linear.predictors - fx)^2)
mod$family$theta    # fixed theta

# negative binomial (unknown theta)
set.seed(1)
y <- rnbinom(n = n, size = 1/2, mu = exp(fx))
mod <- gsm(y ~ x, family = NegBin, knots = 10)
mean((mod$linear.predictors - fx)^2)
mod$family$theta    # estimated theta

# gamma
set.seed(1)
y <- rgamma(n = n, shape = 2, scale = (1 / (2 + fx)) / 2)
mod <- gsm(y ~ x, family = Gamma, knots = 10)
mean((mod$linear.predictors - fx - 2)^2)

# inverse.gaussian (not run; requires statmod)
##set.seed(1)
##y <- statmod::rinvgauss(n = n, mean = sqrt(1 / (2 + fx)), shape = 2)
##mod <- gsm(y ~ x, family = inverse.gaussian, knots = 10)
##mean((mod$linear.predictors - fx - 2)^2)



##########   EXAMPLE 2   ##########
```

```
### 1 continuous and 1 nominal predictor
### additive model

# generate data
n <- 1000
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x) - 1.5
}
fx <- fun(x, z)

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])

# gaussian (default)
set.seed(1)
y <- fx + rnorm(n, sd = 1/sqrt(2))
mod <- gsm(y ~ x + z, knots = knots)
mean((mod$linear.predictors - fx)^2)

# compare to result from sm (they are identical)
mod.sm <- sm(y ~ x + z, knots = knots)
mean((mod$linear.predictors - mod.sm$fitted.values)^2)

# binomial (no weights)
set.seed(1)
y <- rbinom(n = n, size = 1, p = 1 / (1 + exp(-fx)))
mod <- gsm(y ~ x + z, family = binomial, knots = knots)
mean((mod$linear.predictors - fx)^2)

# binomial (w/ weights)
set.seed(1)
w <- as.integer(rep(c(10,20,30,40,50), length.out = n))
y <- rbinom(n = n, size = w, p = 1 / (1 + exp(-fx))) / w
mod <- gsm(y ~ x + z, family = binomial, weights = w, knots = knots)
mean((mod$linear.predictors - fx)^2)

# poisson
set.seed(1)
y <- rpois(n = n, lambda = exp(fx))
mod <- gsm(y ~ x + z, family = poisson, knots = knots)
mean((mod$linear.predictors - fx)^2)

# negative binomial (known theta)
set.seed(1)
y <- rnbinom(n = n, size = 1/2, mu = exp(fx))
mod <- gsm(y ~ x + z, family = NegBin(theta = 1/2), knots = knots)
mean((mod$linear.predictors - fx)^2)
```

```
mod$family$theta    # fixed theta

# negative binomial (unknown theta)
set.seed(1)
y <- rnbinom(n = n, size = 1/2, mu = exp(fx))
mod <- gsm(y ~ x + z, family = NegBin, knots = knots)
mean((mod$linear.predictors - fx)^2)
mod$family$theta    # estimated theta

# gamma
set.seed(1)
y <- rgamma(n = n, shape = 2, scale = (1 / (4 + fx)) / 2)
mod <- gsm(y ~ x + z, family = Gamma, knots = knots)
mean((mod$linear.predictors - fx - 4)^2)

# inverse.gaussian (not run; requires statmod)
##set.seed(1)
##y <- statmod::rinvgauss(n = n, mean = sqrt(1 / (4 + fx)), shape = 2)
##mod <- gsm(y ~ x + z, family = inverse.gaussian, knots = knots)
##mean((mod$linear.predictors - fx - 4)^2)
```

NegBin *Family Function for Negative Binomial*

Description

Creates the functions needed to fit a Negative Binomial generalized smooth model via gsm with or without a known theta parameter. Adapted from the negative.binomial function in the **MASS** package.

Usage

```
NegBin(theta = NULL, link = "log")
```

Arguments

theta	the size parameter for the Negative Binomial distribution. Default of NULL indicates that theta should be estimated from the data.
link	the link function. Must be log, sqrt, identity, or an object of class link-glm (as generated by make.link).

Details

The Negative Binomial distribution has mean μ and variance $\mu + \mu^2/\theta$, where the size parameter θ is the inverse of the dispersion parameter. See NegBinomial for details.

Value

An object of class "family" with the functions and expressions needed to fit the gsm. In addition to the standard values (see family), this also produces the following:

logLik	function to evaluate the log-likelihood
canpar	function to compute the canonical parameter
cumulant	function to compute the cumulant function
theta	the specified theta parameter
fixed.theta	logical specifying if theta was provided

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Venables, W. N. and Ripley, B. D. (1999) Modern Applied Statistics with S-PLUS. Third Edition. Springer.

https://www.rdocumentation.org/packages/MASS/versions/7.3-51.6/topics/negative.binomial

https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/NegBinomial

See Also

gsm for fitting generalized smooth models with Negative Binomial responses

theta.mle for maximum likelihood estimation of theta

Examples

```
# generate data
n <- 1000
x <- seq(0, 1, length.out = n)
fx <- 3 * x + sin(2 * pi * x) - 1.5

# negative binomial (size = 1/2, log link)
set.seed(1)
y <- rnbinom(n = n, size = 1/2, mu = exp(fx))

# fit model (known theta)
mod <- gsm(y ~ x, family = NegBin(theta = 1/2), knots = 10)
mean((mod$linear.predictors - fx)^2)
mod$family$theta   # fixed theta

# fit model (unknown theta)
mod <- gsm(y ~ x, family = NegBin, knots = 10)
mean((mod$linear.predictors - fx)^2)
mod$family$theta   # estimated theta
```

nominal *Nominal Smoothing Spline Basis and Penalty*

Description

Generate the smoothing spline basis and penalty matrix for a nominal spline. This basis and penalty are for an unordered factor.

Usage

```
basis.nom(x, knots, K = NULL, intercept = FALSE, ridge = FALSE)

penalty.nom(x, K = NULL)
```

Arguments

x	Predictor variable (basis) or spline knots (penalty). Factor or integer vector of length n.
knots	Spline knots. Factor or integer vector of length r.
K	Number of levels of x. If NULL, this argument is defined as K = length(unique(x)).
intercept	If TRUE, the first column of the basis will be a column of ones.
ridge	If TRUE, the basis matrix is post-multiplied by the inverse square root of the penalty matrix. See Note and Examples.

Details

Generates a basis function or penalty matrix used to fit nominal smoothing splines.

With an intercept included, the basis function matrix has the form

$$X = [X_0, X_1]$$

where matrix X_0 is an n by 1 matrix of ones, and X_1 is a matrix of dimension n by r.

The X_0 matrix contains the "parametric part" of the basis (i.e., the intercept). The matrix X_1 contains the "nonparametric part" of the basis, which consists of the *reproducing kernel* function

$$\rho(x, y) = \delta_{xy} - 1/K$$

evaluated at all combinations of x and knots. The notation δ_{xy} denotes Kronecker's delta function.

The penalty matrix consists of the *reproducing kernel* function

$$\rho(x, y) = \delta_{xy} - 1/K$$

evaluated at all combinations of x.

Value

Basis: Matrix of dimension c(length(x),df) where df = length(knots) + intercept.

Penalty: Matrix of dimension c(r,r) where r = length(x) is the number of knots.

Note

If the inputs `x` and `knots` are factors, they should have the same levels.

If the inputs `x` and `knots` are integers, the `knots` should be a subset of the input `x`.

If `ridge = TRUE`, the penalty matrix is the identity matrix.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). Smoothing Spline ANOVA Models. 2nd Ed. New York, NY: Springer-Verlag. doi: 10.1007/9781461453697

Helwig, N. E. (2017). Regression with ordered predictors via ordinal smoothing splines. *Frontiers in Applied Mathematics and Statistics, 3*(15), 1-13. doi: 10.3389/fams.2017.00015

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

Helwig, N. E., & Ma, P. (2015). Fast and stable multiple smoothing parameter selection in smoothing spline analysis of variance models with large samples. *Journal of Computational and Graphical Statistics, 24*(3), 715-732. doi: 10.1080/10618600.2014.926819

See Also

See `ordinal` for a basis and penalty for ordered factors.

Examples

```
################   standard parameterization   ################

# generate data
set.seed(0)
n <- 101
x <- factor(sort(rep(LETTERS[1:4], length.out = n)))
knots <- LETTERS[1:3]
eta <- 1:4
y <- eta[x] + rnorm(n, sd = 0.5)

# nominal smoothing spline basis
X <- basis.nom(x, knots, intercept = TRUE)

# nominal smoothing spline penalty
Q <- penalty.nom(knots, K = 4)

# pad Q with zeros (for intercept)
Q <- rbind(0, cbind(0, Q))

# define smoothing parameter
lambda <- 1e-5
```

```
# estimate coefficients
coefs <- solve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta[x] - yhat)^2))



################   ridge parameterization   ######**********

# generate data
set.seed(0)
n <- 101
x <- factor(sort(rep(LETTERS[1:4], length.out = n)))
knots <- LETTERS[1:3]
eta <- 1:4
y <- eta[x] + rnorm(n, sd = 0.5)

# nominal smoothing spline basis
X <- basis.nom(x, knots, intercept = TRUE, ridge = TRUE)

# nominal smoothing spline penalty (ridge)
Q <- diag(rep(c(0, 1), times = c(1, ncol(X) - 1)))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- solve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta[x] - yhat)^2))
```

`ordinal` *Ordinal Smoothing Spline Basis and Penalty*

Description

Generate the smoothing spline basis and penalty matrix for an ordinal spline. This basis and penalty are for an ordered factor.

Usage

```
basis.ord(x, knots, K = NULL, intercept = FALSE, ridge = FALSE)

penalty.ord(x, K = NULL, xlev = NULL)
```

Arguments

x	Predictor variable (basis) or spline knots (penalty). Ordered factor or integer vector of length n.
knots	Spline knots. Ordered factor or integer vector of length r.
K	Number of levels of x. If NULL, this argument is defined as K = length(unique(x)).
xlev	Factor levels of x (for penalty). If NULL, the levels are defined as levels(as.ordered(x)).
intercept	If TRUE, the first column of the basis will be a column of ones.
ridge	If TRUE, the basis matrix is post-multiplied by the inverse square root of the penalty matrix. See Note and Examples.

Details

Generates a basis function or penalty matrix used to fit ordinal smoothing splines.

With an intercept included, the basis function matrix has the form

$$X = [X_0, X_1]$$

where matrix X_0 is an n by 1 matrix of ones, and X_1 is a matrix of dimension n by r. The X_0 matrix contains the "parametric part" of the basis (i.e., the intercept). The matrix X_1 contains the "nonparametric part" of the basis, which consists of the *reproducing kernel* function

$$\rho(x, y) = 1 - (x \vee y) + (1/2K) * (x(x-1) + y(y-1)) + c$$

evaluated at all combinations of x and knots. The notation $(x \vee y)$ denotes the maximum of x and y, and the constant is $c = (K-1)(2K-1)/(6K)$.

The penalty matrix consists of the *reproducing kernel* function

$$\rho(x, y) = 1 - (x \vee y) + (1/2K) * (x(x-1) + y(y-1)) + c$$

evaluated at all combinations of x.

Value

Basis: Matrix of dimension c(length(x),df) where df = length(knots) + intercept.

Penalty: Matrix of dimension c(r,r) where r = length(x) is the number of knots.

Note

If the inputs x and knots are factors, they should have the same levels.

If the inputs x and knots are integers, the knots should be a subset of the input x.

If ridge = TRUE, the penalty matrix is the identity matrix.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). Smoothing Spline ANOVA Models. 2nd Ed. New York, NY: Springer-Verlag. doi: 10.1007/9781461453697

Helwig, N. E. (2017). Regression with ordered predictors via ordinal smoothing splines. *Frontiers in Applied Mathematics and Statistics, 3*(15), 1-13. doi: 10.3389/fams.2017.00015

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations*. doi: 10.4135/9781526421036885885

See Also

See `nominal` for a basis and penalty for unordered factors.

See `polynomial` for a basis and penalty for numeric variables.

Examples

```
#################   standard parameterization   #################

# generate data
set.seed(0)
n <- 101
x <- factor(sort(rep(LETTERS[1:4], length.out = n)))
knots <- LETTERS[1:3]
eta <- 1:4
y <- eta[x] + rnorm(n, sd = 0.5)

# ordinal smoothing spline basis
X <- basis.ord(x, knots, intercept = TRUE)

# ordinal smoothing spline penalty
Q <- penalty.ord(knots, K = 4)

# pad Q with zeros (for intercept)
Q <- rbind(0, cbind(0, Q))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- solve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta[x] - yhat)^2))
```

```
##############   ridge parameterization   ##########

# generate data
set.seed(0)
n <- 101
x <- factor(sort(rep(LETTERS[1:4], length.out = n)))
knots <- LETTERS[1:3]
eta <- 1:4
y <- eta[x] + rnorm(n, sd = 0.5)

# ordinal smoothing spline basis
X <- basis.ord(x, knots, intercept = TRUE, ridge = TRUE)

# ordinal smoothing spline penalty (ridge)
Q <- diag(rep(c(0, 1), times = c(1, ncol(X) - 1)))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- solve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta[x] - yhat)^2))
```

---

plotci *Generic X-Y Plotting with Confidence Intervals*

Description

Modification to the plot function that adds confidence intervals. The CIs can be plotted using polygons (default) or error bars.

Usage

```
plotci(x, y, se, level = 0.95, crit.val = NULL,
       add = FALSE, col = NULL, col.ci = NULL,
       alpha = NULL, bars = NULL, bw = 0.05,
       linkinv = NULL, ...)
```

Arguments

x	a vector of 'x' values (n by 1). If y is missing, the x input can be a list or matrix containing the x, y, and se arguments.
y	a vector of 'y' values (n by 1).
se	a vector of standard error values (n by 1).
level	confidence level for the intervals (between 0 and 1).
crit.val	an optional critical value for the intervals. If provided, the level input is ignored. See Details.
add	a switch controlling whether a new plot should be created (via a call to plot) or if the plot should be added to the current plot (via a call to lines).
col	a character specifying the color for plotting the lines/points.
col.ci	a character specifying the color for plotting the intervals.
alpha	a scalar between 0 and 1 controlling the transparency of the intervals.
bars	a switch controlling whether the intervals should be plotted as bars or polygons.
bw	a positive scalar controlling the bar width. Ignored if bars = FALSE.
linkinv	an inverse link function for the plotting. If provided, the function plots x versus linkinv(y) and the intervals are similarly transformed.
...	extra arguments passed to the plot or lines function.

Details

This function plots x versus y with confidence intervals. The CIs have the form
lwr = y -crit.val * se
upr = y + crit.val * se
where crit.val is the critical value.

If crit.val = NULL, the critival value is determined from the level input as
crit.val <-qnorm(1-(1-level)/2)
where qnorm is the quantile function for the standard normal distribution.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

See Also

This function is used by plot.ss to plot smoothing spline fits.

Examples

```
# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
fx <- 2 + 3 * x + sin(2 * pi * x)
y <- fx + rnorm(n, sd = 0.5)
```

```
# fit smooth model
smod <- sm(y ~ x, knots = 10)

# plot fit with 95% CI polygon
plotci(x, smod$fitted.values, smod$se.fit)

# plot fit with 95% CI bars
plotci(x, smod$fitted.values, smod$se.fit, bars = TRUE)

# plot fit +/- 1 SE
plotci(x, smod$fitted.values, smod$se.fit, crit.val = 1, bars = TRUE)
```

`polynomial` *Polynomial Smoothing Spline Basis and Penalty*

Description

Generate the smoothing spline basis and penalty matrix for a polynomial spline. Derivatives of the smoothing spline basis matrix are supported.

Usage

```
basis.poly(x, knots, m = 2, d = 0, xmin = min(x), xmax = max(x),
           periodic = FALSE, rescale = FALSE, intercept = FALSE,
           bernoulli = TRUE, ridge = FALSE)

penalty.poly(x, m = 2, xmin = min(x), xmax = max(x),
             periodic = FALSE, rescale = FALSE, bernoulli = TRUE)
```

Arguments

`x`	Predictor variable (basis) or spline knots (penalty). Numeric or integer vector of length n.
`knots`	Spline knots. Numeric or integer vector of length r.
`m`	Penalty order. "m=1" for linear smoothing spline, "m=2" for cubic, and "m=3" for quintic.
`d`	Derivative order. "d=0" for smoothing spline basis, "d=1" for 1st derivative of basis, and "d=2" for 2nd derivative of basis.
`xmin`	Minimum value of "x".
`xmax`	Maximum value of "x".
`periodic`	If `TRUE`, the smoothing spline basis is periodic w.r.t. the interval [`xmin`, `xmax`].
`rescale`	If `TRUE`, the nonparametric part of the basis is divided by the average of the reproducing kernel function evaluated at the `knots`.
`intercept`	If `TRUE`, the first column of the basis will be a column of ones.

bernoulli	If TRUE, scaled Bernoulli polynomials are used for the basis and penalty functions.
ridge	If TRUE, the basis matrix is post-multiplied by the inverse square root of the penalty matrix. See Note and Examples.

Details

Generates a basis function or penalty matrix used to fit linear, cubic, and quintic smoothing splines (or evaluate their derivatives).

For non-periodic smoothing splines, the basis function matrix has the form

$$X = [X_0, X_1]$$

where the matrix X_0 is of dimension n by $m-1$ (plus 1 if an intercept is included), and X_1 is a matrix of dimension n by r.

The X_0 matrix contains the "parametric part" of the basis, which includes polynomial functions of x up to degree $m-1$.

The matrix X_1 contains the "nonparametric part" of the basis, which consists of the *reproducing kernel* function

$$\rho(x,y) = \kappa_m(x)\kappa_m(y) + (-1)^{m-1}\kappa_{2m}(|x-y|)$$

evaluated at all combinations of x and knots. The κ_v functions are scaled Bernoulli polynomials.

For periodic smoothing splines, the X_0 matrix only contains the intercept column and the modified reproducing kernel function

$$\rho(x,y) = (-1)^{m-1}\kappa_{2m}(|x-y|)$$

is evaluated for all combinations of x and knots.

For non-periodic smoothing splines, the penalty matrix consists of the *reproducing kernel* function

$$\rho(x,y) = \kappa_m(x)\kappa_m(y) + (-1)^{m-1}\kappa_{2m}(|x-y|)$$

evaluated at all combinations of x. For periodic smoothing splines, the modified reproducing kernel function

$$\rho(x,y) = (-1)^{m-1}\kappa_{2m}(|x-y|)$$

is evaluated for all combinations of x.

If bernoulli = FALSE, the reproducing kernel function is defined as

$$\rho(x,y) = (1/(m-1)!)^2 \int_0^1 (x-u)_+^{m-1}(y-u)_+^{m-1}du$$

where $(.)_+ = \max(.,0)$. This produces the "classic" definition of a smoothing spline, where the function estimate is a piecewise polynomial function with pieces of degree $2m-1$.

Value

Basis: Matrix of dimension c(length(x),df) where df >= length(knots). If the smoothing spline basis is not periodic (default), then the number of columns is df = length(knots) + m -!intercept. For periodic smoothing splines, the basis has m fewer columns.

Penalty: Matrix of dimension c(r,r) where r = length(x) is the number of knots.

Note

Inputs `x` and `knots` should be within the interval [`xmin`, `xmax`].

If `ridge = TRUE`, the penalty matrix is the identity matrix.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). Smoothing Spline ANOVA Models. 2nd Ed. New York, NY: Springer-Verlag. doi: 10.1007/9781461453697

Helwig, N. E. (2017). Regression with ordered predictors via ordinal smoothing splines. *Frontiers in Applied Mathematics and Statistics, 3*(15), 1-13. doi: 10.3389/fams.2017.00015

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

Helwig, N. E., & Ma, P. (2015). Fast and stable multiple smoothing parameter selection in smoothing spline analysis of variance models with large samples. *Journal of Computational and Graphical Statistics, 24*(3), 715-732. doi: 10.1080/10618600.2014.926819

See Also

See `thinplate` for a thin-plate spline basis and penalty.

See `ordinal` for a basis and penalty for ordered factors.

Examples

```
################   standard parameterization   ################

# generate data
set.seed(0)
n <- 101
x <- seq(0, 1, length.out = n)
knots <- seq(0, 0.95, by = 0.05)
eta <- 1 + 2 * x + sin(2 * pi * x)
y <- eta + rnorm(n, sd = 0.5)

# cubic smoothing spline basis
X <- basis.poly(x, knots, intercept = TRUE)

# cubic smoothing spline penalty
Q <- penalty.poly(knots, xmin = min(x), xmax = max(x))

# pad Q with zeros (for intercept and linear effect)
Q <- rbind(0, 0, cbind(0, 0, Q))

# define smoothing parameter
lambda <- 1e-5
```

```
# estimate coefficients
coefs <- solve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta - yhat)^2))

# plot results
plot(x, y)
lines(x, yhat)



################   ridge parameterization   ##########**********

# generate data
set.seed(0)
n <- 101
x <- seq(0, 1, length.out = n)
knots <- seq(0, 0.95, by = 0.05)
eta <- 1 + 2 * x + sin(2 * pi * x)
y <- eta + rnorm(n, sd = 0.5)

# cubic smoothing spline basis
X <- basis.poly(x, knots, intercept = TRUE, ridge = TRUE)

# cubic smoothing spline penalty (ridge)
Q <- diag(rep(c(0, 1), times = c(2, ncol(X) - 2)))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- solve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta - yhat)^2))

# plot results
plot(x, y)
lines(x, yhat)
```

`predict.gsm` *Predict method for Generalized Smooth Model Fits*

Description

predict method for class "gsm".

Usage

```
## S3 method for class 'gsm'
predict(object, newdata = NULL, se.fit = FALSE,
        type = c("link", "response", "terms"),
        terms = NULL, na.action = na.pass,
        intercept = NULL, combine = TRUE, design = FALSE,
        check.newdata = TRUE, ...)
```

Arguments

object	a fit from gsm.
newdata	an optional list or data frame in which to look for variables with which to predict. If omitted, the original data are used.
se.fit	a switch indicating if standard errors are required.
type	type of prediction (link, response, or model term). Can be abbreviated.
terms	which terms to include in the fit. The default of NULL uses all terms. This input **is** used regardless of the type of prediction.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
intercept	a switch indicating if the intercept should be included in the prediction. If NULL (default), the intercept is included in the fit only when type = "r" and terms includes all model terms.
combine	a switch indicating if the parametric and smooth components of the prediction should be combined (default) or returned separately.
design	a switch indicating if the model (design) matrix for the prediction should be returned.
check.newdata	a switch indicating if the newdata should be checked for consistency (e.g., class and range). Ignored if newdata is not provided.
...	additional arguments affecting the prediction produced (currently ignored).

Details

Inspired by the predict.glm function in R's **stats** package.

Produces predicted values, obtained by evaluating the regression function in the frame newdata (which defaults to model.frame(object)). If the logical se.fit is TRUE, standard errors of the predictions are calculated.

If newdata is omitted the predictions are based on the data used for the fit. Regardless of the newdata argument, how cases with missing values are handled is determined by the na.action argument. If na.action = na.omit omitted cases will not appear in the predictions, whereas if na.action = na.exclude they will appear (in predictions and standard errors), with value NA.

Similar to the glm function, setting type = "terms" returns a matrix giving the predictions for each of the requested model terms. Unlike the glm function, this function allows for predictions using any subset of the model terms. Specifically, the predictions (on both the link and response scale) will only include the requested terms, which makes it possible to obtain estimates (and standard errors) for subsets of model terms. In this case, the newdata only needs to contain data for the subset of variables that are requested in terms.

Value

Default use returns a vector of predictions. Otherwise the form of the output will depend on the combination of argumments: se.fit, type, combine, and design.

type = "link":
When se.fit = FALSE and design = FALSE, the output will be the predictions on the link scale. When se.fit = TRUE or design = TRUE, the output is a list with components fit, se.fit (if requested), and X (if requested).

type = "response":
When se.fit = FALSE and design = FALSE, the output will be the predictions on the data scale. When se.fit = TRUE or design = TRUE, the output is a list with components fit, se.fit (if requested), and X (if requested).

type = "terms":
When se.fit = FALSE and design = FALSE, the output will be the predictions for each term on the link scale. When se.fit = TRUE or design = TRUE, the output is a list with components fit, se.fit (if requested), and X (if requested).

Regardless of the type, setting combine = FALSE decomposes the requested result(s) into the **p**arametric and **s**mooth contributions.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.glm.html

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik, 31*, 377-403. doi: 10.1007/BF01404567

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

See Also

gsm

Examples

```
# generate data
set.seed(1)
n <- 1000
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x + mu[zi]*pi/4)
}
fx <- fun(x, z)
y <- rbinom(n = n, size = 1, p = 1 / (1 + exp(-fx)))

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])

# fit gsm with specified knots (tprk = TRUE)
gsm.ssa <- gsm(y ~ x * z, family = binomial, knots = knots)
pred <- predict(gsm.ssa)
term <- predict(gsm.ssa, type = "terms")
mean((gsm.ssa$linear.predictors - pred)^2)
mean((gsm.ssa$linear.predictors - rowSums(term) - attr(term, "constant"))^2)

# fit gsm with specified knots (tprk = FALSE)
gsm.gam <- gsm(y ~ x * z, family = binomial, knots = knots, tprk = FALSE)
pred <- predict(gsm.gam)
term <- predict(gsm.gam, type = "terms")
mean((gsm.gam$linear.predictors - pred)^2)
mean((gsm.gam$linear.predictors - rowSums(term) - attr(term, "constant"))^2)
```

`predict.sm` *Predict method for Smooth Model Fits*

Description

`predict` method for class "sm".

Usage

```
## S3 method for class 'sm'
predict(object, newdata = NULL, se.fit = FALSE,
        interval = c("none", "confidence", "prediction"),
        level = 0.95, type = c("response", "terms"),
        terms = NULL, na.action = na.pass,
        intercept = NULL, combine = TRUE, design = FALSE,
        check.newdata = TRUE, ...)
```

Arguments

object	a fit from sm.
newdata	an optional list or data frame in which to look for variables with which to predict. If omitted, the original data are used.
se.fit	a switch indicating if standard errors are required.
interval	type of interval calculation. Can be abbreviated.
level	tolerance/confidence level.
type	type of prediction (response or model term). Can be abbreviated.
terms	which terms to include in the fit. The default of NULL uses all terms. This input **is** used regardless of the type of prediction.
na.action	function determining what should be done with missing values in newdata. The default is to predict NA.
intercept	a switch indicating if the intercept should be included in the prediction. If NULL (default), the intercept is included in the fit only when type = "r" and terms includes all model terms.
combine	a switch indicating if the parametric and smooth components of the prediction should be combined (default) or returned separately.
design	a switch indicating if the model (design) matrix for the prediction should be returned.
check.newdata	a switch indicating if the newdata should be checked for consistency (e.g., class and range). Ignored if newdata is not provided.
...	additional arguments affecting the prediction produced (currently ignored).

Details

Inspired by the predict.lm function in R's **stats** package.

Produces predicted values, obtained by evaluating the regression function in the frame newdata (which defaults to model.frame(object)). If the logical se.fit is TRUE, standard errors of the predictions are calculated. Setting intervals specifies computation of confidence or prediction (tolerance) intervals at the specified level, sometimes referred to as narrow vs. wide intervals.

If newdata is omitted the predictions are based on the data used for the fit. Regardless of the newdata argument, how cases with missing values are handled is determined by the na.action argument. If na.action = na.omit omitted cases will not appear in the predictions, whereas if na.action = na.exclude they will appear (in predictions, standard errors or interval limits), with value NA.

Similar to the lm function, setting type = "terms" returns a matrix giving the predictions for each of the requested model terms. Unlike the lm function, this function allows for predictions using any subset of the model terms. Specifically, when type = "response" the predictions will only include the requested terms, which makes it possible to obtain estimates (and standard errors and intervals) for subsets of model terms. In this case, the newdata only needs to contain data for the subset of variables that are requested in terms.

Value

Default use returns a vector of predictions. Otherwise the form of the output will depend on the combination of argumments: `se.fit`, `interval`, `type`, `combine`, and `design`.

`type = "response"`:
When `se.fit = FALSE` and `design = FALSE`, the output will be the predictions (possibly with `lwr` and `upr` interval bounds). When `se.fit = TRUE` or `design = TRUE`, the output is a list with components `fit`, `se.fit` (if requested), and `X` (if requested).

`type = "terms"`:
When `se.fit = FALSE` and `design = FALSE`, the output will be the predictions for each term (possibly with `lwr` and `upr` interval bounds). When `se.fit = TRUE` or `design = TRUE`, the output is a list with components `fit`, `se.fit` (if requested), and `X` (if requested).

Regardless of the `type`, setting `combine = FALSE` decomposes the requested result(s) into the **p**arametric and **s**mooth contributions.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.lm.html

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik, 31*, 377-403. doi: 10.1007/BF01404567

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

See Also

`sm`

Examples

```
# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x + mu[zi]*pi/4)
}
fx <- fun(x, z)
y <- fx + rnorm(n, sd = 0.5)
```

```
# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])

# fit sm with specified knots
smod <- sm(y ~ x * z, knots = knots)

# get model "response" predictions
fit <- predict(smod)
mean((smod$fitted.values - fit)^2)

# get model "terms" predictions
trm <- predict(smod, type = "terms")
attr(trm, "constant")
head(trm)
mean((smod$fitted.values - rowSums(trm) - attr(trm, "constant"))^2)

# get predictions with "newdata" (= the original data)
fit <- predict(smod, newdata = data.frame(x = x, z = z))
mean((fit - smod$fitted.values)^2)

# get predictions and standard errors
fit <- predict(smod, se.fit = TRUE)
mean((fit$fit - smod$fitted.values)^2)
mean((fit$se.fit - smod$se.fit)^2)

# get 99% confidence interval
fit <- predict(smod, interval = "c", level = 0.99)
head(fit)

# get 99% prediction interval
fit <- predict(smod, interval = "p", level = 0.99)
head(fit)

# get predictions only for x main effect
fit <- predict(smod, newdata = data.frame(x = x),
               se.fit = TRUE, terms = "x")
plotci(x, fit$fit, fit$se.fit)

# get predictions only for each group
fit.a <- predict(smod, newdata = data.frame(x = x, z = "a"), se.fit = TRUE)
fit.b <- predict(smod, newdata = data.frame(x = x, z = "b"), se.fit = TRUE)
fit.c <- predict(smod, newdata = data.frame(x = x, z = "c"), se.fit = TRUE)

# plot results (truth as dashed line)
plotci(x = x, y = fit.a$fit, se = fit.a$se.fit,
       col = "red", col.ci = "pink", ylim = c(-6, 6))
lines(x, fun(x, rep(1, n)), lty = 2, col = "red")
plotci(x = x, y = fit.b$fit, se = fit.b$se.fit,
       col = "blue", col.ci = "cyan", add = TRUE)
lines(x, fun(x, rep(2, n)), lty = 2, col = "blue")
```

```
plotci(x = x, y = fit.c$fit, se = fit.c$se.fit,
       col = "darkgreen", col.ci = "lightgreen", add = TRUE)
lines(x, fun(x, rep(3, n)), lty = 2, col = "darkgreen")

# add legends
legend("bottomleft", legend = c("Truth", "Estimate", "CI"),
       lty = c(2, 1, NA), lwd = c(1, 2, NA),
       col = c("black", "black","gray80"),
       pch = c(NA, NA, 15), pt.cex = 2, bty = "n")
legend("bottomright", legend = letters[1:3],
       lwd = 2, col = c("red", "blue", "darkgreen"), bty = "n")
```

`predict.ss` *Predict method for Smoothing Spline Fits*

Description

`predict` method for class "ss".

Usage

```
## S3 method for class 'ss'
predict(object, x, deriv = 0, se.fit = TRUE, ...)
```

Arguments

`object`	a fit from `ss`.
`x`	the new values of x.
`deriv`	integer; the order of the derivative required.
`se.fit`	a switch indicating if standard errors are required.
`...`	additional arguments affecting the prediction produced (currently ignored).

Details

Inspired by the `predict.smooth.spline` function in R's **stats** package.

Value

A list with components

`x`	The input `x`.
`y`	The fitted values or derivatives at x.
`se`	The standard errors of the fitted values or derivatives (if requested).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.smooth.spline.html

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik, 31*, 377-403. doi: 10.1007/BF01404567

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

See Also

ss

Examples

```
# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
fx <- 2 + 3 * x + sin(2 * pi * x)
y <- fx + rnorm(n, sd = 0.5)

# GCV selection (default)
ss.GCV <- ss(x, y, nknots = 10)

# get predictions and SEs (at design points)
fit <- predict(ss.GCV, x = x)
head(fit)

# compare to original fit
mean((fit$y - ss.GCV$y)^2)

# plot result (with default 95% CI)
plotci(fit)

# estimate first derivative
d1 <- 3 + 2 * pi * cos(2 * pi * x)
fit <- predict(ss.GCV, x = x, deriv = 1)
head(fit)

# plot result (with default 95% CI)
plotci(fit)
lines(x, d1, lty = 2)   # truth
```

psolve *Pseudo-Solve a System of Equations*

Description

This generic function solves the equation `a %*% x = b` for `x`, where `b` can be either a vector or a matrix. This implementation is similar to `solve`, but uses a pseudo-inverse if the system is computationally singular.

Usage

```
psolve(a, b, tol)
```

Arguments

`a`	a rectangular numeric matrix containing the coefficients of the linear system.
`b`	a numeric vector or matrix giving the right-hand side(s) of the linear system. If missing, `b` is taken to be an identity matrix and solve will return the (pseudo-)inverse of `a`.
`tol`	the tolerance for detecting linear dependencies in the columns of a. The default is `.Machine$double.eps`.

Details

If `a` is a symmetric matrix, `eigen` is used to compute the (pseudo-)inverse. This assumes that `a` is a positive semi-definite matrix. Otherwise `svd` is used to compute the (pseudo-)inverse for rectangular matrices.

Value

If `b` is missing, returns the (pseudo-)inverse of `a`. Otherwise returns `psolve(a) %*% b`.

Note

The pseudo-inverse is calculated by inverting the eigen/singular values that are greater than the first value multiplied by `tol * min(dim(a))`.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Moore, E. H. (1920). On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society, 26*, 394-395. doi: 10.1090/S0002990419200332 27

Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society, 51(3)*, 406-413. doi: 10.1017/S0305004100030401

See Also

solve

Examples

```
# generate X
set.seed(0)
X <- matrix(rnorm(100), 20, 5)
X <- cbind(X, rowSums(X))

# pseudo-inverse of X  (dim = 5 by 20)
Xinv <- psolve(X)

# pseudo-inverse of crossprod(X)  (dim = 5 by 5)
XtXinv <- psolve(crossprod(X))
```

---

sm *Fit a Smooth Model*

Description

Fits a semi- or nonparametric regression model with the smoothing parameter selected via one of eight methods: GCV, OCV, GACV, ACV, REML, ML, AIC, or BIC.

Usage

```
sm(formula, data, weights, types = NULL, tprk = TRUE, knots = NULL,
   update = TRUE, df, spar = NULL, lambda = NULL, control = list(),
   method = c("GCV", "OCV", "GACV", "ACV", "REML", "ML", "AIC", "BIC"))
```

Arguments

formula	Object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. Uses the same syntax as lm and glm.
data	Optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which sm is called.
weights	Optional vector of weights to be used in the fitting process. If provided, weighted least squares is used. Defaults to all 1.
types	Named list giving the type of smooth to use for each predictor. If NULL, the type is inferred from the data. See "Types of Smooths" section for details.
tprk	Logical specifying how to parameterize smooth models with multiple predictors. If TRUE (default), a **t**ensor **p**roduct **r**eproducing **k**ernel function is used to represent the function. If FALSE, a tensor product of marginal kernel functions is used to represent the function. See the "Multiple Smooths" section for details.

`knots` Spline knots for the estimation of the nonparametric effects. For models with multiple predictors, the knot specification will depend on the `tprk` input. See the "Choosing Knots" section for details

`update` If `TRUE`, steps 1-2 of Gu and Wahba's (1991) algorithm 3.2 are used to update the "extra" smoothing parameters. If `FALSE`, only step 1 of algorithm 3.2 is used, so each effect is given equal influence on the penalty. Only applicable when multiple smooth terms are included.

`df` Equivalent degrees of freedom (trace of the smoother matrix). Must be in $[m, n]$ where m is the number of columns of the null space basis function matrix X, and n is the number of observations.

`spar` Smoothing parameter. Typically (but not always) in the range $(0, 1]$. If specified `lambda = 256^(3*(spar-1))`.

`lambda` Computational smoothing parameter. This value is weighted by n to form the penalty coefficient (see Details). Ignored if `spar` is provided.

`control` Optional list with named components controlling the root finding when the smoothing parameter spar is computed, i.e., missing or NULL, see below.

Note that spar is only searched for in the interval $[lower, upper]$.

lower: lower bound for spar; defaults to 0.

upper: upper bound for spar; defaults to 1.

tol: the absolute precision (**tol**erance) used by `optimize`; defaults to 1e-8.

`method` Method for selecting the smoothing parameter. Ignored if `lambda` is provided.

Details

Letting $f_i = f(x_i)$ with $x_i = (x_{i1}, \ldots, x_{ip})$, the function is represented as

$$f = X\beta + Z\gamma$$

where the basis functions in X span the null space (i.e., parametric effects), and Z contains the kernel function(s) of the contrast space (i.e., nonparametric effects) evaluated at all combinations of observed data points and knots. The vectors β and γ contain unknown basis function coefficients.

Letting $M = (X, Z)$ and $\theta = (\beta', \gamma')'$, the penalized least squares problem has the form

$$(y - M\theta)'W(y - M\theta) + n\lambda\gamma'Q\gamma$$

where W is a diagonal matrix containg the weights, and Q is the penalty matrix. The optimal coefficients are the solution to

$$(M'WM + n\lambda P)\theta = M'Wy$$

where P is the penalty matrix Q augmented with zeros corresponding to the β in θ.

Value

An object of class "sm" with components:

`fitted.values` the fitted values, i.e., predictions.

se.fit	the standard errors of the fitted values.
sse	the sum-of-squared errors.
cv.crit	the cross-validation criterion.
df	the estimated degrees of freedom (Df) for the fit model.
nsdf	the degrees of freedom (Df) for the null space.
r.squared	the observed coefficient of multiple determination.
sigma	the estimate of the error standard deviation.
logLik	the log-likelihood (if method is REML or ML).
aic	Akaike's Information Criterion (if method is AIC).
bic	Bayesian Information Criterion (if method is BIC).
spar	the value of spar computed or given, i.e., $s = 1 + \log_{256}(\lambda)/3$
lambda	the value of λ corresponding to spar, i.e., $\lambda = 256^{3*(s-1)}$.
penalty	the smoothness penalty $\gamma' Q \gamma$.
coefficients	the basis function coefficients used for the fit model.
cov.sqrt	the square-root of the covariance matrix of coefficients. Note: tcrossprod(cov.sqrt) reconstructs the covariance matrix.
specs	a list with information used for prediction purposes: **knots** the spline knots used for each predictor. **thetas** the "extra" tuning parameters used to weight the penalties. **xrng** the ranges of the predictor variables. **xlev** the factor levels of the predictor variables (if applicable). **tprk** logical controlling the formation of tensor product smooths.
data	the data used to fit the model.
types	the type of smooth used for each predictor.
terms	the terms included in the fit model.
method	the method used for smoothing parameter selection. Will be NULL if lambda was provided.
formula	the formula specifying the fit model.
call	the matched call.

Methods

The smoothing parameter can be selected using one of eight methods:
Generalized Cross-Validation (GCV)
Ordinary Cross-Validation (OCV)
Generalized Approximate Cross-Validation (GACV)
Approximate Cross-Validation (ACV)
Restricted Maximum Likelihood (REML)
Maximum Likelihood (ML)
Akaike's Information Criterion (AIC)
Bayesian Information Criterion (BIC)

Types of Smooths

The following codes specify the spline types:

par	Parametric effect (factor, integer, or numeric).
nom	Nominal smoothing spline (unordered factor).
ord	Ordinal smoothing spline (ordered factor).
lin	Linear smoothing spline (integer or numeric).
cub	Cubic smoothing spline (integer or numeric).
qui	Quintic smoothing spline (integer or numeric).
per	Periodic smoothing spline (integer or numeric).
sph	Spherical spline (matrix with $d = 3$ columns).
tps	Thin-plate spline (matrix with $d \geq 1$ columns).

For finer control of some specialized spline types:

per.lin	Linear periodic spline ($m = 1$).
per.cub	Cubic periodic spline ($m = 2$).
per.qui	Quintic periodic spline ($m = 3$).
sph.lin	Linear spherical spline ($m = 1$).
sph.cub	Cubic spherical spline ($m = 2$).
sph.qui	Quintic spherical spline ($m = 3$).
tps.lin	Linear thin-plate spline ($m = 1$).
tps.cub	Cubic thin-plate spline ($m = 2$).
tps.qui	Quintic thin-plate spline ($m = 3$).

For details on the spline kernel functions, see `basis.nom` (nominal), `basis.ord` (ordinal), `basis.poly` (polynomial), `basis.sph` (spherical), and `basis.tps` (thin-plate).

Choosing Knots

If `tprk = TRUE`, the four options for the `knots` input include:

1. a scalar giving the total number of knots to sample
2. a vector of integers indexing which rows of data are the knots
3. a list with named elements giving the marginal knot values for each predictor (to be combined via `expand.grid`)
4. a list with named elements giving the knot values for each predictor (requires the same number of knots for each predicto

If `tprk = FALSE`, the three options for the `knots` input include:

1. a scalar giving the common number of knots for each continuous predictor
2. a list with named elements giving the number of marginal knots for each predictor
3. a list with named elements giving the marginal knot values for each predictor

Multiple Smooths

Suppose `formula = y ~ x1 + x2` so that the model contains additive effects of two predictor variables.

The k-th predictor's marginal effect can be denoted as

$$f_k = X_k \beta_k + Z_k \gamma_k$$

where X_k is the n by m_k null space basis function matrix, and Z_k is the n by r_k contrast space basis function matrix.

If `tprk = TRUE`, the null space basis function matrix has the form $X = [1, X_1, X_2]$ and the contrast space basis function matrix has the form

$$Z = \theta_1 Z_1 + \theta_2 Z_2$$

where the θ_k are the "extra" smoothing parameters. Note that Z is of dimension n by $r = r_1 = r_2$.

If `tprk = FALSE`, the null space basis function matrix has the form $X = [1, X_1, X_2]$, and the contrast space basis function matrix has the form

$$Z = [\theta_1 Z_1, \theta_2 Z_2]$$

where the θ_k are the "extra" smoothing parameters. Note that Z is of dimension n by $r = r_1 + r_2$.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik, 31*, 377-403. doi: 10.1007/BF01404567

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Gu, C. and Wahba, G. (1991). Minimizing GCV/GML scores with multiple smoothing parameters via the Newton method. *SIAM Journal on Scientific and Statistical Computing, 12(2)*, 383-398. doi: 10.1137/0912021

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), SAGE Research Methods Foundations. doi: 10.4135/9781526421036885885

Helwig, N. E. (2020+). Spectrally sparse nonparametric regression via elastic net regularized smoothers. *Journal of Computational and Graphical Statistics*. doi: 10.1080/10618600.2020.1806855

See Also

`summary.sm` for summarizing `sm` objects.

`predict.sm` for predicting from `sm` objects.

`ss` for fitting a smoothing spline with a single predictor (Gaussian response).

`gsm` for fitting generalized smooth models with multiple predictors of mixed types (non-Gaussian response).

Examples

```
##########   EXAMPLE 1   ##########
### 1 continuous predictor

# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
fx <- 2 + 3 * x + sin(2 * pi * x)
y <- fx + rnorm(n, sd = 0.5)

# fit sm with 10 knots (tprk = TRUE)
sm.ssa <- sm(y ~ x, knots = 10)

# fit sm with 10 knots (tprk = FALSE)
sm.gam <- sm(y ~ x, knots = 10, tprk = FALSE)

# print both results (note: they are identical)
sm.ssa
sm.gam

# summarize both results (note: they are identical)
summary(sm.ssa)
summary(sm.gam)

# compare true MSE values (note: they are identical)
mean( ( fx - sm.ssa$fit )^2 )
mean( ( fx - sm.gam$fit )^2 )



##########   EXAMPLE 2   ##########
### 1 continuous and 1 nominal predictor
### additive model

# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x)
}
fx <- fun(x, z)
y <- fx + rnorm(n, sd = 0.5)

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])
```

```
# fit sm with specified knots (tprk = TRUE)
sm.ssa <- sm(y ~ x + z, knots = knots)

# fit sm with specified knots (tprk = FALSE)
sm.gam <- sm(y ~ x + z, knots = knots, tprk = FALSE)

# print both results (note: they are identical)
sm.ssa
sm.gam

# summarize both results (note: they are almost identical)
summary(sm.ssa)
summary(sm.gam)

# compare true MSE values (note: they are identical)
mean( ( fx - sm.ssa$fit )^2 )
mean( ( fx - sm.gam$fit )^2 )



##########   EXAMPLE 3   ##########
### 1 continuous and 1 nominal predictor
### interaction model

# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x + mu[zi]*pi/4)
}
fx <- fun(x, z)
y <- fx + rnorm(n, sd = 0.5)

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])

# fit sm with specified knots (tprk = TRUE)
sm.ssa <- sm(y ~ x * z, knots = knots)

# fit sm with specified knots (tprk = FALSE)
sm.gam <- sm(y ~ x * z, knots = knots, tprk = FALSE)

# print both results (note: they are slightly different)
sm.ssa
sm.gam
```

```
# summarize both results (note: they are slightly different)
summary(sm.ssa)
summary(sm.gam)

# compare true MSE values (note: they are slightly different)
mean( ( fx - sm.ssa$fit )^2 )
mean( ( fx - sm.gam$fit )^2 )



##########   EXAMPLE 4   ##########
### 4 continuous predictors
### additive model

# generate data
set.seed(1)
n <- 100
fun <- function(x){
  sin(pi*x[,1]) + sin(2*pi*x[,2]) + sin(3*pi*x[,3]) + sin(4*pi*x[,4])
}
data <- as.data.frame(replicate(4, runif(n)))
colnames(data) <- c("x1v", "x2v", "x3v", "x4v")
fx <- fun(data)
y <- fx + rnorm(n)

# define marginal knots
knots <- list(x1v = quantile(data$x1v, probs = seq(0, 1, length.out = 10)),
              x2v = quantile(data$x2v, probs = seq(0, 1, length.out = 10)),
              x3v = quantile(data$x3v, probs = seq(0, 1, length.out = 10)),
              x4v = quantile(data$x4v, probs = seq(0, 1, length.out = 10)))

# define ssa knot indices
knots.indx <- c(bin.sample(data$x1v, nbin = 10, index.return = TRUE)$ix,
                bin.sample(data$x2v, nbin = 10, index.return = TRUE)$ix,
                bin.sample(data$x3v, nbin = 10, index.return = TRUE)$ix,
                bin.sample(data$x4v, nbin = 10, index.return = TRUE)$ix)

# fit sm with specified knots (tprk = TRUE)
sm.ssa <- sm(y ~ x1v + x2v + x3v + x4v, data = data, knots = knots.indx)

# fit sm with specified knots (tprk = FALSE)
sm.gam <- sm(y ~ x1v + x2v + x3v + x4v, data = data, knots = knots, tprk = FALSE)

# print both results (note: they are slightly different)
sm.ssa
sm.gam

# summarize both results (note: they are slightly different)
summary(sm.ssa)
summary(sm.gam)

# compare true MSE values (note: they are slightly different)
mean( ( fx - sm.ssa$fit )^2 )
```

```
mean( ( fx - sm.gam$fit )^2 )
```

---

spherical *Spherical Spline Basis and Penalty*

Description

Generate the smoothing spline basis and penalty matrix for a spherical spline. This basis is designed for a 3D predictor where the values are points on a sphere.

Usage

```
basis.sph(x, knots, m = 2, rescale = TRUE, intercept = FALSE, ridge = FALSE)

penalty.sph(x, m = 2, rescale = TRUE)
```

Arguments

x	Predictor variables (basis) or spline knots (penalty). Matrix of dimension n by 3.
knots	Spline knots. Matrix of dimension r by 3.
m	Penalty order. "m=1" for linear spherical spline, "m=2" for cubic, and "m=3" for quintic.
rescale	If TRUE, the nonparametric part of the basis is divided by the average of the reproducing kernel function evaluated at the knots.
intercept	If TRUE, the first column of the basis will be a column of ones.
ridge	If TRUE, the basis matrix is post-multiplied by the inverse square root of the penalty matrix. See Note and Examples.

Details

Generates a basis function or penalty matrix used to fit linear, cubic, and spherical splines.

With an intercept included, the basis function matrix has the form

$$X = [X_0, X_1]$$

where matrix X_0 is an n by 1 matrix of ones, and X_1 is a matrix of dimension n by r.

The X_0 matrix contains the "parametric part" of the basis (i.e., the intercept).

The matrix X_1 contains the "nonparametric part" of the basis, which consists of the *reproducing kernel* function

$$\rho(x, y) = 1 + [s_{2m}(x.y) - \alpha_m]/\beta_m$$

evaluated at all combinations of x and knots. Note that $\alpha_m = 1/(2m+1)$ and $\beta_m = 2\pi(2m)!$ are constants, $s_{2m}(.)$ is the spherical spline semi-kernel function, and $x.y$ denote the inner product between x and y (see References).

The penalty matrix consists of the *reproducing kernel* function

$$\rho(x, y) = 1 + [s_{2m}(x.y) - \alpha_m]/\beta_m$$

evaluated at all combinations of `x`.

Value

Basis: Matrix of dimension `c(length(x),df)` where `df = nrow(knots) + intercept`.

Penalty: Matrix of dimension `c(r,r)` where `r = nrow(x)` is the number of knots.

Note

The inputs `x` and `knots` must have the same dimension.

If `ridge = TRUE`, the penalty matrix is the identity matrix.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). Smoothing Spline ANOVA Models. 2nd Ed. New York, NY: Springer-Verlag. doi: 10.1007/9781461453697

Wahba, G (1981). Spline interpolation and smoothing on the sphere. *SIAM Journal on Scientific Computing, 2*(1), 5-16. doi: 10.1137/0902002

See Also

See `thinplate` for a thin-plate spline basis and penalty.

Examples

```
################   standard parameterization   ########**######

# function with three spherical predictors
set.seed(0)
n <- 1000
myfun <- function(x){
  sin(pi*x[,1]) + cos(2*pi*x[,2]) + cos(pi*x[,3])
  }
x <- cbind(runif(n), runif(n), runif(n)) - 0.5
x <- t(apply(x, 1, function(x) x / sqrt(sum(x^2))))
eta <- myfun(x)
y <- eta + rnorm(n, sd = 0.5)
knots <- x[1:100,]

# cubic spherical spline basis
X <- basis.sph(x, knots, intercept = TRUE)

# cubic spherical spline penalty
```

```
Q <- penalty.sph(knots)

# pad Q with zeros (for intercept)
Q <- rbind(0, cbind(0, Q))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- psolve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta - yhat)^2))



################   ridge parameterization   ################

# function with three spherical predictors
set.seed(0)
n <- 1000
myfun <- function(x){
  sin(pi*x[,1]) + cos(2*pi*x[,2]) + cos(pi*x[,3])
  }
x <- cbind(runif(n), runif(n), runif(n)) - 0.5
x <- t(apply(x, 1, function(x) x / sqrt(sum(x^2))))
eta <- myfun(x)
y <- eta + rnorm(n, sd = 0.5)
knots <- x[1:100,]

# cubic spherical spline basis
X <- basis.sph(x, knots, intercept = TRUE, ridge = TRUE)

# cubic spherical spline penalty (ridge)
Q <- diag(rep(c(0, 1), times = c(1, ncol(X) - 1)))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- psolve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta - yhat)^2))
```

ss *Fit a Smoothing Spline*

Description

Fits a smoothing spline with the smoothing parameter selected via one of eight methods: GCV, OCV, GACV, ACV, REML, ML, AIC, or BIC.

Usage

```
ss(x, y = NULL, w = NULL, df, spar = NULL, lambda = NULL,
   method = c("GCV", "OCV", "GACV", "ACV", "REML", "ML", "AIC", "BIC"),
   m = 2L, periodic = FALSE, all.knots = FALSE, nknots = .nknots.smspl,
   knots = NULL, keep.data = TRUE, df.offset = 0, penalty = 1,
   control.spar = list(), tol = 1e-6 * IQR(x), bernoulli = TRUE)
```

Arguments

x	Predictor vector of length n. Can also input a list or a two-column matrix specifying x and y.
y	Response vector of length n. If y is missing or NULL, the responses are assumed to be specified by x, with x the index vector.
w	Weights vector of length n. Defaults to all 1.
df	Equivalent degrees of freedom (trace of the smoother matrix). Must be in $[m, nx]$, where nx is the number of unique x values, see below.
spar	Smoothing parameter. Typically (but not always) in the range $(0, 1]$. If specified lambda = 256^(3*(spar-1)).
lambda	Computational smoothing parameter. This value is weighted by n to form the penalty coefficient (see Details). Ignored if spar is provided.
method	Method for selecting the smoothing parameter. Ignored if spar or lambda is provided.
m	Penalty order (integer). The penalty functional is the integrated squared m-th derivative of the function. Defaults to $m = 2$, which is a cubic smoothing spline. Set $m = 1$ for a linear smoothing spline or $m = 3$ for a quintic smoothing spline.
periodic	Logical. If TRUE, the estimated function $f(x)$ is constrained to be periodic, i.e., $f(a) = f(b)$ where $a = \min(x)$ and $b = \max(x)$.
all.knots	If TRUE, all distinct points in x are used as knots. If FALSE (default), a sequence knots is placed at the quantiles of the unique x values; in this case, the input nknots specifies the number of knots in the sequence. Ignored if the knot values are input using the knots argument.
nknots	Positive integer or function specifying the number of knots. Ignored if either all.knots = TRUE or the knot values are input using the knots argument.
knots	Vector of knot values for the spline. Should be unique and within the range of the x values (to avoid a warning).

keep.data	Logical. If TRUE, the original data as a part of the output object.
df.offset	Allows the degrees of freedom to be increased by df.offset in the GCV criterion.
penalty	The coefficient of the penalty for degrees of freedom in the GCV criterion.
control.spar	Optional list with named components controlling the root finding when the smoothing parameter spar is computed, i.e., missing or NULL, see below. **Note** that spar is only searched for in the interval $[lower, upper]$. **lower:** lower bound for spar; defaults to 0. **upper:** upper bound for spar; defaults to 1. **tol:** the absolute precision (**tol**erance) used by optimize; defaults to 1e-8.
tol	Tolerance for same-ness or uniqueness of the x values. The values are binned into bins of size tol and values which fall into the same bin are regarded as the same. Must be strictly positive (and finite).
bernoulli	If TRUE, scaled Bernoulli polynomials are used for the basis and penalty functions. If FALSE, produces the "classic" definition of a smoothing spline, where the function estimate is a piecewise polynomial function with pieces of degree $2m - 1$. See polynomial for details.

Details

Inspired by the smooth.spline function in R's **stats** package.

Neither x nor y are allowed to containing missing or infinite values.

The x vector should contain at least $2m$ distinct values. 'Distinct' here is controlled by tol: values which are regarded as the same are replaced by the first of their values and the corresponding y and w are pooled accordingly.

Unless lambda has been specified instead of spar, the computational λ used (as a function of spar) is $\lambda = 256^(3 * (spar - 1))$.

If spar and lambda are missing or NULL, the value of df is used to determine the degree of smoothing. If df is missing as well, the specified method is used to determine λ.

Letting $f_i = f(x_i)$, the function is represented as

$$f = X\beta + Z\gamma$$

where the basis functions in X span the null space (i.e., functions with m-th derivative of zero), and Z contains the reproducing kernel function of the contrast space evaluated at all combinations of observed data points and knots, i.e., $Z[i, j] = \rho(x_i, k_j)$ where ρ is the kernel function and k_j is the j-th knot. The vectors β and γ contain unknown basis function coefficients. Letting $M = (X, Z)$ and $\theta = (\beta', \gamma')'$, the penalized least squares problem has the form

$$(y - M\theta)'W(y - M\theta) + n\lambda\gamma'Q\gamma$$

where W is a diagonal matrix containg the weights, and Q is the penalty matrix. Note that $Q[i, j] = \rho(k_i, k_j)$ contains the reproducing kernel function evaluated at all combinations of knots. The optimal coefficients are the solution to

$$(M'WM + n\lambda P)\theta = M'Wy$$

where P is the penalty matrix Q augmented with zeros corresponding to the β in θ.

Value

An object of class "ss" with components:

x	the distinct x values in increasing order; see Note.
y	the fitted values corresponding to x.
w	the weights used at the unique values of x.
yin	the y values used at the unique y values.
tol	the tol argument (whose default depends on x).
data	only if keep.data = TRUE: itself a list with components x, y and w (if applicable). These are the original $(x_i, y_i, w_i), i = 1, \ldots, n$, values where data$x may have repeated values and hence be longer than the above x component; see details.
lev	leverages, the diagonal values of the smoother matrix.
cv.crit	cross-validation score.
pen.crit	the penalized criterion, a non-negative number; simply the (weighted) residual sum of squares (RSS).
crit	the criterion value minimized in the underlying df2lambda function. When df is provided, the criterion is $[tr(S_\lambda) - df]^2$.
df	equivalent degrees of freedom used.
spar	the value of spar computed or given, i.e., $s = 1 + \log_{256}(\lambda)/3$
lambda	the value of λ corresponding to spar, i.e., $\lambda = 256^{3*(s-1)}$.
fit	list for use by predict.ss, with components **n:** number of observations. **knot:** the knot sequence. **nk:** number of coefficients (# knots plus m). **coef:** coefficients for the spline basis used. **min, range:** numbers giving the corresponding quantities of x **m:** spline penalty order (same as input m) **periodic:** is spline periodic? **cov.sqrt** square root of covariance matrix of coef such that tcrossprod(coef) reconstructs the covariance matrix. **weighted** were weights w used in fitting? **bernoulli** were Bernoulli polynomials used in fitting?
call	the matched call.
sigma	estimated error standard deviation.
logLik	log-likelihood (if method is REML or ML).
aic	Akaike's Information Criterion (if method is AIC).
bic	Bayesian Information Criterion (if method is BIC).
penalty	smoothness penalty $\gamma' Q \gamma$, which is the integrated squared m-th derivative of the estimated function $f(x)$.
method	smoothing parameter selection method. Will be NULL if df, spar, or lambda is provided.

Methods

The smoothing parameter can be selected using one of eight methods:
Generalized Cross-Validation (GCV)
Ordinary Cross-Validation (OCV)
Generalized Approximate Cross-Validation (GACV)
Approximate Cross-Validation (ACV)
Restricted Maximum Likelihood (REML)
Maximum Likelihood (ML)
Akaike's Information Criterion (AIC)
Bayesian Information Criterion (BIC)

Note

The number of unique x values, nx, are determined by the tol argument, equivalently to

```
nx <-sum(!duplicated( round((x -mean(x)) / tol) ))
```

In this case where not all unique x values are used as knots, the result is not a smoothing spline in the strict sense, but very close unless a small smoothing parameter (or large `df`) is used.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/smooth.spline.html

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik, 31*, 377-403. doi: 10.1007/BF01404567

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

Helwig, N. E. (2020+). Spectrally sparse nonparametric regression via elastic net regularized smoothers. *Journal of Computational and Graphical Statistics*. doi: 10.1080/10618600.2020.1806855

Wahba, G. (1985). A comparison of GCV and GML for choosing the smoothing parameters in the generalized spline smoothing problem. *The Annals of Statistics, 4*, 1378-1402. doi: 10.1214/aos/1176349743

See Also

`summary.ss` for summarizing `ss` objects.

`predict.ss` for predicting from `ss` objects.

`sm` for fitting smooth models with multiple predictors of mixed types (Gaussian response).

`gsm` for fitting generalized smooth models with multiple predictors of mixed types (non-Gaussian response).

Examples

```
# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
fx <- 2 + 3 * x + sin(2 * pi * x)
y <- fx + rnorm(n, sd = 0.5)

# GCV selection (default)
ss.GCV <- ss(x, y, nknots = 10)
ss.GCV

# OCV selection
ss.OCV <- ss(x, y, method = "OCV", nknots = 10)
ss.OCV

# GACV selection
ss.GACV <- ss(x, y, method = "GACV", nknots = 10)
ss.GACV

# ACV selection
ss.ACV <- ss(x, y, method = "ACV", nknots = 10)
ss.ACV

# ML selection
ss.ML <- ss(x, y, method = "ML", nknots = 10)
ss.ML

# REML selection
ss.REML <- ss(x, y, method = "REML", nknots = 10)
ss.REML

# AIC selection
ss.AIC <- ss(x, y, method = "AIC", nknots = 10)
ss.AIC

# BIC selection
ss.BIC <- ss(x, y, method = "BIC", nknots = 10)
ss.BIC

# compare results
mean( ( fx - ss.GCV$y )^2 )
mean( ( fx - ss.OCV$y )^2 )
mean( ( fx - ss.GACV$y )^2 )
mean( ( fx - ss.ACV$y )^2 )
mean( ( fx - ss.ML$y )^2 )
mean( ( fx - ss.REML$y )^2 )
mean( ( fx - ss.AIC$y )^2 )
mean( ( fx - ss.BIC$y )^2 )

# plot results
plot(x, y)
```

```
rlist <- list(ss.GCV, ss.OCV, ss.GACV, ss.ACV,
              ss.REML, ss.ML, ss.AIC, ss.BIC)
for(j in 1:length(rlist)){
   lines(rlist[[j]], lwd = 2, col = j)
}
```

---

summary *Summary methods for Fit Models*

Description

summary methods for object classes "gsm", "sm", and "ss".

Usage

```
## S3 method for class 'gsm'
summary(object, ...)

## S3 method for class 'sm'
summary(object, ...)

## S3 method for class 'ss'
summary(object, ...)

## S3 method for class 'summary.gsm'
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'summary.sm'
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'summary.ss'
print(x, digits = max(3, getOption("digits") - 3),
      signif.stars = getOption("show.signif.stars"), ...)
```

Arguments

object	an object of class "gsm" output by the gsm function, "sm" output by the sm function, or "ss" output by the ss function
x	an object of class "summary.gsm" output by the summary.gsm function, "summary.sm" output by the summary.sm function, or "summary.ss" output by the summary.ss function.
digits	the minimum number of significant digits to be printed in values.
signif.stars	logical. If TRUE, 'significance stars' are printed for each coefficient.
...	additional arguments affecting the summary produced (currently ignored).

Details

Summary includes information for assessing the statistical and practical significance of the model terms.

Statistical inference is conducted via (approximate) frequentist chi-square tests using the Bayesian interpretation of a smoothing spline (Nychka, 1988; Wahba, 1983).

With multiple smooth terms included in the model, the inferential results may (and likely will) differ slightly depending on the `tprk` argument (when using the `gsm` and `sm` functions).

If significance testing is of interest, the `tprk = FALSE` option may be desirable, given that this allows for unique basis function coefficients for each model term.

In all cases, the inferential results are based on a (pseudo) F or chi-square statistic which fails to consider the uncertainty of the smoothing parameter estimation.

Value

`residuals`	the deviance residuals.
`fstatistic`	the F statistic for testing all effects (parametric and smooth).
`dev.expl`	the explained deviance.
`p.table`	the coefficient table for (approximate) inference on the parametric terms.
`s.table`	the coefficient table for (approximate) inference on the smooth terms.
`dispersion`	the estimate of the dispersion parameter.
`r.squared`	the observed coefficient of multiple determination.
`adj.r.squared`	the adjusted coefficient of multiple determination.
`kappa`	the collinearity indices. A value of 1 indicates no collinearity, and higher values indicate more collinearity of a given term with other model terms.
`pi`	the importance indices. Larger values indicate more importance, and the values satisfy `sum(pi) = 1`. Note that elements of `pi` can be negative.
`call`	the original function call.
`family`	the specified family (for gsm objects).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), *SAGE Research Methods Foundations.* doi: 10.4135/9781526421036885885

Nychka, D. (1988). Bayesian confience intervals for smoothing splines. *Journal of the American Statistical Association, 83(404)*, 1134-1143. doi: 10.2307/2290146

Wahba, G. (1983). Bayesian "confidence intervals" for the cross-validated smoothing spline. *Journal of the Royal Statistical Society. Series B, 45(1)*, 133-150. doi: 10.1111/j.25176161.1983.tb01239.x

See Also

gsm, sm, and ss

Examples

```
### Example 1: gsm

# generate data
set.seed(1)
n <- 1000
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x + mu[zi]*pi/4)
}
fx <- fun(x, z)
y <- rbinom(n = n, size = 1, p = 1 / (1 + exp(-fx)))

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])

# fit sm with specified knots (tprk = TRUE)
gsm.ssa <- gsm(y ~ x * z, family = binomial, knots = knots)
summary(gsm.ssa)

# fit sm with specified knots (tprk = FALSE)
gsm.gam <- gsm(y ~ x * z, family = binomial, knots = knots, tprk = FALSE)
summary(gsm.gam)


### Example 2: sm

# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x + mu[zi]*pi/4)
}
fx <- fun(x, z)
y <- fx + rnorm(n, sd = 0.5)

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
```

```
              z = letters[1:3])

# fit sm with specified knots (tprk = TRUE)
sm.ssa <- sm(y ~ x * z, knots = knots)
summary(sm.ssa)

# fit sm with specified knots (tprk = FALSE)
sm.gam <- sm(y ~ x * z, knots = knots, tprk = FALSE)
summary(sm.gam)


### Example 3: ss

# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
fx <- 2 + 3 * x + sin(2 * pi * x)
y <- fx + rnorm(n, sd = 0.5)

# regular smoothing spline
ss.reg <- ss(x, y, nknots = 10)
summary(ss.reg)
```

`theta.mle` *MLE of Theta for Negative Binomial*

Description

Computes the maximum likelihood estimate of the size (theta) parameter for the Negative Binomial distribution via a Newton-Raphson algorithm.

Usage

```
theta.mle(y, mu, theta, wt = 1,
          maxit = 100, maxth = .Machine$double.xmax,
          tol = .Machine$double.eps^0.5)
```

Arguments

`y`	response vector
`mu`	mean vector
`theta`	initial theta (optional)
`wt`	weight vector
`maxit`	max number of iterations
`maxth`	max possible value of `theta`
`tol`	convergence tolerance

Details

Based on the `glm.nb` function in the **MASS** package. If `theta` is missing, the initial estimate of theta is given by

```
theta <-1 / mean(wt * (y / mu -1)^2)
```

which is motivated by the method of moments estimator for the dispersion parameter in a quasi-Poisson model.

Value

Returns estimated theta with attributes

`SE`	standard error estimate
`iter`	number of iterations

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Venables, W. N. and Ripley, B. D. (1999) Modern Applied Statistics with S-PLUS. Third Edition. Springer.

https://www.rdocumentation.org/packages/MASS/versions/7.3-51.6/topics/negative.binomial

https://www.rdocumentation.org/packages/MASS/versions/7.3-51.6/topics/glm.nb

See Also

`NegBin` for details on the Negative Binomial distribution

Examples

```
# generate data
n <- 1000
x <- seq(0, 1, length.out = n)
fx <- 3 * x + sin(2 * pi * x) - 1.5
mu <- exp(fx)

# simulate negative binomial data
set.seed(1)
y <- rnbinom(n = n, size = 1/2, mu = mu)

# estimate theta
theta.mle(y, mu)
```

`thinplate` *Thin-Plate Spline Basis and Penalty*

Description

Generate the smoothing spline basis and penalty matrix for a thin-plate spline.

Usage

```
basis.tps(x, knots, m = 2, rk = TRUE, intercept = FALSE, ridge = FALSE)

penalty.tps(x, m = 2, rk = TRUE)
```

Arguments

`x`	Predictor variables (basis) or spline knots (penalty). Numeric or integer vector of length n, or matrix of dimension n by p.
`knots`	Spline knots. Numeric or integer vector of length r, or matrix of dimension r by p.
`m`	Penalty order. "m=1" for linear thin-plate spline, "m=2" for cubic, and "m=3" for quintic. Must satisfy $2m > p$.
`rk`	If true (default), the reproducing kernel parameterization is used. Otherwise, the standard thin-plate basis is returned.
`intercept`	If `TRUE`, the first column of the basis will be a column of ones.
`ridge`	If `TRUE`, the basis matrix is post-multiplied by the inverse square root of the penalty matrix. Only applicable if `rk = TRUE`. See Note and Examples.

Details

Generates a basis function or penalty matrix used to fit linear, cubic, and quintic thin-plate splines.

The basis function matrix has the form

$$X = [X_0, X_1]$$

where the matrix `X_0` is of dimension n by $M - 1$ (plus 1 if an intercept is included) where $M = \binom{p+m-1}{p}$, and `X_1` is a matrix of dimension n by r.

The `X_0` matrix contains the "parametric part" of the basis, which includes polynomial functions of the columns of `x` up to degree $m - 1$ (and potentially interactions).

The matrix `X_1` contains the "nonparametric part" of the basis.

If `rk = TRUE`, the matrix `X_1` consists of the *reproducing kernel* function

$$\rho(x, y) = (I - P_x)(I - P_y)E(|x - y|)$$

evaluated at all combinations of `x` and `knots`. Note that P_x and P_y are projection operators, $|.|$ denotes the Euclidean distance, and the TPS semi-kernel is defined as

$$E(z) = \alpha z^{2m-p} \log(z)$$

if p is even and

$$E(z) = \beta z^{2m-p}$$

otherwise, where α and β are positive constants (see References).

If `rk = FALSE`, the matrix `X_1` contains the TPS semi-kernel $E(.)$ evaluated at all combinations of `x` and `knots`. Note: the TPS semi-kernel is *not* positive (semi-)definite, but the projection is.

If `rk = TRUE`, the penalty matrix consists of the *reproducing kernel* function

$$\rho(x, y) = (I - P_x)(I - P_y)E(|x - y|)$$

evaluated at all combinations of `x`. If `rk = FALSE`, the penalty matrix contains the TPS semi-kernel $E(.)$ evaluated at all combinations of `x`.

Value

Basis: Matrix of dimension `c(length(x),df)` where `df = nrow(as.matrix(knots)) + choose(p + m -1,p) -!intercept` and `p = ncol(as.matrix(x))`.

Penalty: Matrix of dimension `c(r,r)` where `r = nrow(as.matrix(x))` is the number of knots.

Note

The inputs `x` and `knots` must have the same dimension.

If `rk = TRUE` and `ridge = TRUE`, the penalty matrix is the identity matrix.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). Smoothing Spline ANOVA Models. 2nd Ed. New York, NY: Springer-Verlag. doi: 10.1007/9781461453697

Helwig, N. E. (2017). Regression with ordered predictors via ordinal smoothing splines. *Frontiers in Applied Mathematics and Statistics, 3*(15), 1-13. doi: 10.3389/fams.2017.00015

Helwig, N. E., & Ma, P. (2015). Fast and stable multiple smoothing parameter selection in smoothing spline analysis of variance models with large samples. *Journal of Computational and Graphical Statistics, 24*(3), 715-732. doi: 10.1080/10618600.2014.926819

See Also

See `polynomial` for a basis and penalty for numeric variables.

See `spherical` for a basis and penalty for spherical variables.

Examples

```
#################   standard parameterization   #################

# generate data
set.seed(0)
n <- 101
x <- seq(0, 1, length.out = n)
knots <- seq(0, 0.95, by = 0.05)
eta <- 1 + 2 * x + sin(2 * pi * x)
y <- eta + rnorm(n, sd = 0.5)

# cubic thin-plate spline basis
X <- basis.tps(x, knots, intercept = TRUE)

# cubic thin-plate spline penalty
Q <- penalty.tps(knots)

# pad Q with zeros (for intercept and linear effect)
Q <- rbind(0, 0, cbind(0, 0, Q))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- psolve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta - yhat)^2))

# plot results
plot(x, y)
lines(x, yhat)



#################   ridge parameterization   ##################

# generate data
set.seed(0)
n <- 101
x <- seq(0, 1, length.out = n)
knots <- seq(0, 0.95, by = 0.05)
eta <- 1 + 2 * x + sin(2 * pi * x)
y <- eta + rnorm(n, sd = 0.5)

# cubic thin-plate spline basis
X <- basis.tps(x, knots, intercept = TRUE, ridge = TRUE)

# cubic thin-plate spline penalty (ridge)
```

```
Q <- diag(rep(c(0, 1), times = c(2, ncol(X) - 2)))

# define smoothing parameter
lambda <- 1e-5

# estimate coefficients
coefs <- psolve(crossprod(X) + n * lambda * Q) %*% crossprod(X, y)

# estimate eta
yhat <- X %*% coefs

# check rmse
sqrt(mean((eta - yhat)^2))

# plot results
plot(x, y)
lines(x, yhat)
```

---

varimp *Variable Importance*

Description

Computes variable importance indices for terms of a smooth model.

Usage

```
varimp(object, combine = TRUE)
```

Arguments

object	an object of class "sm" output by the sm function or an object of class "gsm" output by the gsm function.
combine	a switch indicating if the parametric and smooth components of the importance should be combined (default) or returned separately.

Details

Suppose that the function can be written as

$$\eta = \eta_0 + \eta_1 + \eta_2 + ... + \eta_p$$

where η_0 is a constant (intercept) term, and η_j denotes the j-th effect function, which is assumed to have mean zero. Note that η_j could be a main or interaction effect function for all $j = 1, ..., p$.

The variable importance index for the j-th effect term is defined as

$$\pi_j = (\eta_j^\top \eta_*)/(\eta_*^\top \eta_*)$$

where $\eta_* = \eta_1 + \eta_2 + ... + \eta_p$. Note that $\sum_{j=1}^{p} \pi_j = 1$ but there is no guarantee that $\pi_j > 0$.

If all π_j are non-negative, then π_j gives the proportion of the model's R-squared that can be accounted for by the j-th effect term. Thus, values of π_j closer to 1 indicate that η_j is more important, whereas values of π_j closer to 0 (including negative values) indicate that η_j is less important.

Value

If `combine = TRUE`, returns a named vector containing the importance indices for each effect function (in `object$terms`).

If `combine = FALSE`, returns a data frame where the first column gives the importance indices for the parametric components and the second column gives the importance indices for the smooth (nonparametric) components.

Note

When `combine = FALSE`, importance indices will be equal to zero for non-existent components of a model term. For example, a `nominal` effect does not have a parametric component, so the `$p` component of the importance index for a nominal effect will be zero.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). Smoothing spline ANOVA models, 2nd edition. New York: Springer. doi: 10.1007/9781461453697

Helwig, N. E. (2020). Multiple and Generalized Nonparametric Regression. In P. Atkinson, S. Delamont, A. Cernat, J. W. Sakshaug, & R. A. Williams (Eds.), SAGE Research Methods Foundations. doi: 10.4135/9781526421036885885

See Also

See `summary.sm` for more thorough summaries of smooth models.

See `summary.gsm` for more thorough summaries of generalized smooth models.

Examples

```
##########   EXAMPLE 1   ##########
### 1 continuous and 1 nominal predictor

# generate data
set.seed(1)
n <- 100
x <- seq(0, 1, length.out = n)
z <- factor(sample(letters[1:3], size = n, replace = TRUE))
fun <- function(x, z){
  mu <- c(-2, 0, 2)
```

```
  zi <- as.integer(z)
  fx <- mu[zi] + 3 * x + sin(2 * pi * x)
}
fx <- fun(x, z)
y <- fx + rnorm(n, sd = 0.5)

# define marginal knots
probs <- seq(0, 0.9, by = 0.1)
knots <- list(x = quantile(x, probs = probs),
              z = letters[1:3])

# fit correct (additive) model
sm.add <- sm(y ~ x + z, knots = knots)

# fit incorrect (interaction) model
sm.int <- sm(y ~ x * z, knots = knots)

# true importance indices
eff <- data.frame(x = 3 * x + sin(2 * pi * x), z = c(-2, 0, 2)[as.integer(z)])
eff <- scale(eff, scale = FALSE)
fstar <- rowSums(eff)
colSums(eff * fstar) / sum(fstar^2)

# estimated importance indices
varimp(sm.add)
varimp(sm.int)



##########   EXAMPLE 2   ##########
### 4 continuous predictors
### additive model

# generate data
set.seed(1)
n <- 100
fun <- function(x){
  sin(pi*x[,1]) + sin(2*pi*x[,2]) + sin(3*pi*x[,3]) + sin(4*pi*x[,4])
}
data <- as.data.frame(replicate(4, runif(n)))
colnames(data) <- c("x1v", "x2v", "x3v", "x4v")
fx <- fun(data)
y <- fx + rnorm(n)

# define ssa knot indices
knots.indx <- c(bin.sample(data$x1v, nbin = 10, index.return = TRUE)$ix,
                bin.sample(data$x2v, nbin = 10, index.return = TRUE)$ix,
                bin.sample(data$x3v, nbin = 10, index.return = TRUE)$ix,
                bin.sample(data$x4v, nbin = 10, index.return = TRUE)$ix)

# fit correct (additive) model
sm.add <- sm(y ~ x1v + x2v + x3v + x4v, data = data, knots = knots.indx)
```

```
# fit incorrect (interaction) model
sm.int <- sm(y ~ x1v * x2v + x3v + x4v, data = data, knots = knots.indx)

# true importance indices
eff <- data.frame(x1v = sin(pi*data[,1]), x2v = sin(2*pi*data[,2]),
                  x3v = sin(3*pi*data[,3]), x4v = sin(4*pi*data[,4]))
eff <- scale(eff, scale = FALSE)
fstar <- rowSums(eff)
colSums(eff * fstar) / sum(fstar^2)

# estimated importance indices
varimp(sm.add)
varimp(sm.int)
```

Index