Package 'bigsplines'

February 3, 2017

Type Package

Title Smoothing Splines for Large Samples

Version 1.1-0

Date 2017-02-01

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

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

Depends quadprog

Imports stats, graphics, grDevices

Description Fits smoothing spline regression models using scalable algorithms designed for large samples. Seven marginal spline types are supported: linear, cubic, different cubic, cubic periodic, cubic thin-plate, ordinal, and nominal. Random effects and parametric effects are also supported. Response can be Gaussian or non-Gaussian: Binomial, Poisson, Gamma, Inverse Gaussian, or Negative Binomial.

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-02-03 14:32:57

R topics documented:

bigsplines-package 2
bigspline 3
bigssa 8
bigssg 14
bigssp 21
bigtps 26
binsamp 30
imagebar 32
makessa 34
makessg 37
makessp 39
ordspline 42

plotbar 44
plotci 46
predict.bigspline 48
predict.bigssa 50
predict.bigssg 54
predict.bigssp 57
predict.bigtps 61
predict.ordspline 63
print 65
ssBasis 67
summary 69

Index **73**

`bigsplines-package` *Smoothing Splines for Large Samples*

Description

Fits smoothing spline regression models using scalable algorithms designed for large samples. Six marginal spline types are supported: cubic, different cubic, cubic periodic, cubic thin-plate, ordinal, and nominal. Random effects and parametric predictors are also supported. Response can be Gaussian or non-Gaussian: Binomial, Poisson, Gamma, Inverse Gaussian, or Negative Binomial.

Details

The function `bigspline` fits one-dimensional cubic smoothing splines (unconstrained or periodic). The function `bigssa` fits Smoothing Spline Anova (SSA) models (Gaussian data). The function `bigssg` fits Generalized Smoothing Spline Anova (GSSA) models (non-Gaussian data). The function `bigssp` is for fitting Smoothing Splines with Parametric effects (semi-parametric regression). The function `bigtps` fits one-, two-, and three-dimensional cubic thin-plate splines. There are corresponding predict, print, and summary functions for these methods.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

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

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

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*, 383-398.

Gu, C. and Xiang, D. (2001). Cross-validating non-Gaussian data: Generalized approximate cross-validation revisited. *Journal of Computational and Graphical Statistics, 10*, 581-591.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
# See examples for bigspline, bigssa, bigssg, bigssp, and bigtps
```

`bigspline` *Fits Smoothing Spline*

Description

Given a real-valued response vector $\mathbf{y} = \{y_i\}_{n\times 1}$ and a real-valued predictor vector $\mathbf{x} = \{x_i\}_{n\times 1}$ with $a \leq x_i \leq b \; \forall i$, a smoothing spline model has the form

$$y_i = \eta(x_i) + e_i$$

where y_i is the i-th observation's respone, x_i is the i-th observation's predictor, η is an unknown smooth function relating the response and predictor, and $e_i \sim \mathrm{N}(0, \sigma^2)$ is iid Gaussian error.

Usage

```
bigspline(x,y,type="cub",nknots=30,rparm=0.01,xmin=min(x),
          xmax=max(x),alpha=1,lambdas=NULL,se.fit=FALSE,
          rseed=1234,knotcheck=TRUE)
```

Arguments

`x`	Predictor vector.
`y`	Response vector. Must be same length as `x`.
`type`	Type of spline for `x`. Options include `type="lin"` for linear, `type="cub"` for cubic, `type="cub0"` for different cubic, and `type="per"` for cubic periodic. See Spline Types section.
`nknots`	Scalar giving maximum number of knots to bin-sample. Use more knots for more jagged functions.
`rparm`	Rounding parameter for `x`. Use `rparm=NA` to fit unrounded solution. Rounding parameter must be in interval (0,1].

xmin	Minimum x value (i.e., a). Used to transform data to interval [0,1].
xmax	Maximum x value (i.e., b). Used to transform data to interval [0,1].
alpha	Manual tuning parameter for GCV score. Using alpha=1 gives unbaised esitmate. Using a larger alpha enforces a smoother estimate.
lambdas	Vector of global smoothing parameters to try. Default estimates smoothing parameter that minimizes GCV score.
se.fit	Logical indicating if the standard errors of fitted values should be estimated.
rseed	Random seed. Input to set.seed to reproduce same knots when refitting same model. Use rseed=NULL to generate a different sample of knots each time.
knotcheck	If TRUE, only unique knots are used (for stability).

Details

To estimate η I minimize the penalized least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \eta(x_i))^2 + \lambda \int [\ddot{\eta}(x)]^2 dx$$

where $\ddot{\eta}$ denotes the second derivative of η and $\lambda \geq 0$ is a smoothing parameter that controls the trade-off between fitting and smoothing the data.

Default use of the function estimates λ by minimizing the GCV score:

$$\text{GCV}(\lambda) = \frac{n\|(\mathbf{I}_n - \mathbf{S}_\lambda)\mathbf{y}\|^2}{[n - \text{tr}(\mathbf{S}_\lambda)]^2}$$

where $\mathbf{I}_n$ is the identity matrix and $\mathbf{S}_\lambda$ is the smoothing matrix (see Computational Details).

Using the rounding parameter input rparm can greatly speed-up and stabilize the fitting for large samples. When rparm is used, the spline is fit to a set of unique data points after rounding; the unique points are determined using the efficient algorithm described in Helwig (2013). For typical cases, I recommend using rparm=0.01, but smaller rounding parameters (e,g., rparm=0.001) may be needed for particularly jagged functions (or when x has outliers).

Value

fitted.values	Vector of fitted values corresponding to the original data points in x (if rparm=NA) or the rounded data points in xunique (if rparm is used).
se.fit	Vector of standard errors of fitted.values (if input se.fit=TRUE).
x	Predictor vector (same as input).
y	Response vector (same as input).
type	Type of spline that was used.
xunique	Unique elements of x after rounding (if rparm is used).
yunique	Mean of y for unique elements of x after rounding (if rparm is used).
funique	Vector giving frequency of each element of xunique (if rparm is used).
sigma	Estimated error standard deviation, i.e., $\hat{\sigma}$.

`ndf`	Data frame with two elements: `n` is total sample size, and `df` is effective degrees of freedom of fit model (trace of smoothing matrix).
`info`	Model fit information: vector containing the GCV, multiple R-squared, AIC, and BIC of fit model (assuming Gaussian error).
`xrng`	Predictor range: `xrng=c(xmin,xmax)`.
`myknots`	Bin-sampled spline knots used for fit.
`rparm`	Rounding parameter for `x` (same as input).
`lambda`	Optimal smoothing parameter.
`coef`	Spline basis function coefficients.
`coef.csqrt`	Matrix square-root of covariace matrix of `coef`. Use `tcrossprod(coef.csqrt)` to get covariance matrix of `coef`.

Warnings

Cubic and cubic periodic splines transform the predictor to the interval [0,1] before fitting. So input `xmin` must be less than or equal to `min(x)`, and input `xmax` must be greater than or equal to `max(x)`.

When using rounding parameters, output `fitted.values` corresponds to unique rounded predictor scores in output `xunique`. Use `predict.bigspline` function to get fitted values for full `y` vector.

Computational Details

According to smoothing spline theory, the function η can be approximated as

$$\eta(x) = d_0 + d_1\phi_1(x) + \sum_{h=1}^{q} c_h \rho(x, x_h^*)$$

where the ϕ_1 is a linear function, ρ is the reproducing kernel of the contrast (nonlinear) space, and $\{x_h^*\}_{h=1}^q$ are the selected spline knots.

This implies that the penalized least-squares functional can be rewritten as

$$\|\mathbf{y} - \mathbf{K}\mathbf{d} - \mathbf{J}\mathbf{c}\|^2 + n\lambda \mathbf{c}'\mathbf{Q}\mathbf{c}$$

where $\mathbf{K} = \{\phi(x_i)\}_{n\times 2}$ is the null space basis function matrix, $\mathbf{J} = \{\rho(x_i, x_h^*)\}_{n\times q}$ is the contrast space basis funciton matrix, $\mathbf{Q} = \{\rho(x_g^*, x_h^*)\}_{q\times q}$ is the penalty matrix, and $\mathbf{d} = (d_0, d_1)'$ and $\mathbf{c} = (c_1, \ldots, c_q)'$ are the unknown basis function coefficients.

Given the smoothing parameter λ, the optimal basis function coefficients have the form

$$\begin{pmatrix} \hat{\mathbf{d}} \\ \hat{\mathbf{c}} \end{pmatrix} = \begin{pmatrix} \mathbf{K}'\mathbf{K} & \mathbf{K}'\mathbf{J} \\ \mathbf{J}'\mathbf{K} & \mathbf{J}'\mathbf{J} + n\lambda\mathbf{Q} \end{pmatrix}^{\dagger} \begin{pmatrix} \mathbf{K}' \\ \mathbf{J}' \end{pmatrix} \mathbf{y}$$

where $(\cdot)^{\dagger}$ denotes the pseudoinverse of the input matrix.

Given the optimal coefficients, the fitted values are given by $\hat{\mathbf{y}} = \mathbf{K}\hat{\mathbf{d}} + \mathbf{J}\hat{\mathbf{c}} = \mathbf{S}_\lambda \mathbf{y}$, where

$$\mathbf{S}_\lambda = \begin{pmatrix} \mathbf{K} & \mathbf{J} \end{pmatrix} \begin{pmatrix} \mathbf{K}'\mathbf{K} & \mathbf{K}'\mathbf{J} \\ \mathbf{J}'\mathbf{K} & \mathbf{J}'\mathbf{J} + n\lambda\mathbf{Q} \end{pmatrix}^{\dagger} \begin{pmatrix} \mathbf{K}' \\ \mathbf{J}' \end{pmatrix}$$

is the smoothing matrix, which depends on λ.

Spline Types

For a linear spline (`type="lin"`) with $x \in [0,1]$, the needed functions are

$$\phi_1(x) = 0 \qquad \text{and} \qquad \rho(x,z) = k_1(x)k_1(z) + k_2(|x-z|)$$

where $k_1(x) = x - 0.5$, $k_2(x) = \frac{1}{2}\left(k_1^2(x) - \frac{1}{12}\right)$; in this case $\mathbf{K} = \mathbf{1}_n$ and $\mathbf{d} = d_0$.

For a cubic spline (`type="cub"`) with $x \in [0,1]$, the needed functions are

$$\phi_1(x) = k_1(x) \qquad \text{and} \qquad \rho(x,z) = k_2(x)k_2(z) - k_4(|x-z|)$$

where k_1 and k_2 are defined above, and $k_4(x) = \frac{1}{24}\left(k_1^4(x) - \frac{k_1^2(x)}{2} + \frac{7}{240}\right)$.

For a different cubic spline (`type="cub0"`) with $x \in [0,1]$, the needed functions are

$$\phi_1(x) = x \qquad \text{and} \qquad \rho(x,z) = (x \wedge z)^2[3(x \vee z) - (x \wedge z)]/6$$

where $(x \wedge z) = \min(x,z)$ and $(x \vee z) = \max(x,z)$.

Note that `type="cub"` and `type="cub0"` use different definitions of the averaging operator in the null space. The overall spline estimates should be the same (up to approximation accuracy), but the null and constrast space effect functions will differ (see `predict.bigspline`). See Helwig (2013) and Gu (2013) for a further discussion of polynomial splines.

For a periodic cubic spline (`type="per"`) with $x \in [0,1]$, the needed functions are

$$\phi_1(x) = 0 \qquad \text{and} \qquad \rho(x,z) = -k_4(|x-z|)$$

where $k_4(x)$ is defined as it was for `type="cub"`; in this case $\mathbf{K} = \mathbf{1}_n$ and $\mathbf{d} = d_0$.

Note

The spline is estimated using penalized least-squares, which does not require the Gaussian error assumption. However, the spline inference information (e.g., standard errors and fit information) requires the Gaussian error assumption.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define relatively smooth function
set.seed(773)
myfun <- function(x){ sin(2*pi*x) }
x <- runif(10^6)
y <- myfun(x) + rnorm(10^6)

# linear, cubic, different cubic, and periodic splines
linmod <- bigspline(x,y,type="lin")
linmod
cubmod <- bigspline(x,y)
cubmod
cub0mod <- bigspline(x,y,type="cub0")
cub0mod
permod <- bigspline(x,y,type="per")
permod


##########   EXAMPLE 2   ##########

# define more jagged function
set.seed(773)
myfun <- function(x){ 2*x + cos(4*pi*x) }
x <- runif(10^6)*4
y <- myfun(x) + rnorm(10^6)

# try different numbers of knots
r1mod <- bigspline(x,y,nknots=20)
crossprod( myfun(r1mod$xunique) - r1mod$fitted )/length(r1mod$fitted)
r2mod <- bigspline(x,y,nknots=30)
crossprod( myfun(r2mod$xunique) - r2mod$fitted )/length(r2mod$fitted)
r3mod <- bigspline(x,y,nknots=40)
crossprod( myfun(r3mod$xunique) - r3mod$fitted )/length(r3mod$fitted)


##########   EXAMPLE 3   ##########

# define more jagged function
set.seed(773)
myfun <- function(x){ 2*x + cos(4*pi*x) }
x <- runif(10^6)*4
y <- myfun(x) + rnorm(10^6)

# try different rounding parameters
r1mod <- bigspline(x,y,rparm=0.05)
crossprod( myfun(r1mod$xunique) - r1mod$fitted )/length(r1mod$fitted)
r2mod <- bigspline(x,y,rparm=0.02)
crossprod( myfun(r2mod$xunique) - r2mod$fitted )/length(r2mod$fitted)
r3mod <- bigspline(x,y,rparm=0.01)
```

```
crossprod( myfun(r3mod$xunique) - r3mod$fitted )/length(r3mod$fitted)
```

---

bigssa *Fits Smoothing Spline ANOVA Models*

Description

Given a real-valued response vector $\mathbf{y} = \{y_i\}_{n\times 1}$, a Smoothing Spline Anova (SSA) has the form

$$y_i = \eta(\mathbf{x}_i) + e_i$$

where y_i is the i-th observation's respone, $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ is the i-th observation's nonparametric predictor vector, η is an unknown smooth function relating the response and nonparametric predictors, and $e_i \sim \mathrm{N}(0, \sigma^2)$ is iid Gaussian error. Function can fit additive models, and also allows for 2-way and 3-way interactions between any number of predictors (see Details and Examples).

Usage

```
bigssa(formula,data=NULL,type=NULL,nknots=NULL,rparm=NA,
       lambdas=NULL,skip.iter=TRUE,se.fit=FALSE,rseed=1234,
       gcvopts=NULL,knotcheck=TRUE,gammas=NULL,weights=NULL,
       random=NULL,remlalg=c("FS","NR","EM","none"),remliter=500,
       remltol=10^-4,remltau=NULL)
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted (see Details and Examples for more information).
data	Optional data frame, list, or environment containing the variables in formula. Or an object of class "makessa", which is output from makessa.
type	List of smoothing spline types for predictors in formula (see Details). Options include type="cub" for cubic, type="cub0" for another cubic, type="per" for cubic periodic, type="tps" for cubic thin-plate, type="ord" for ordinal, and type="nom" for nominal.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of data to use as knots.
rparm	List of rounding parameters for each predictor. See Details.
lambdas	Vector of global smoothing parameters to try. Default lambdas=10^-c(9:0).
skip.iter	Logical indicating whether to skip the iterative smoothing parameter update. Using skip.iter=FALSE should provide a more optimal solution, but the fitting time may be substantially longer. See Skip Iteration section.
se.fit	Logical indicating if the standard errors of the fitted values should be estimated.
rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.

gcvopts	Control parameters for optimization. List with 3 elements: (a) maxit: maximum number of algorithm iterations, (b) gcvtol: covergence tolerance for iterative GCV update, and (c) alpha: tuning parameter for GCV minimization. Default: gcvopts=list(maxit=5,gcvtol=10^-5,alpha=1)
knotcheck	If TRUE, only unique knots are used (for stability).
gammas	List of initial smoothing parameters for each predictor. See Details.
weights	Vector of positive weights for fitting (default is vector of ones).
random	Adds random effects to model (see Random Effects section).
remlalg	REML algorithm for estimating variance components (see Random Effects section). Input is ignored if random=NULL.
remliter	Maximum number of iterations for REML estimation of variance components. Input is ignored if random=NULL.
remltol	Convergence tolerance for REML estimation of variance components. Input is ignored if random=NULL.
remltau	Initial estimate of variance parameters for REML estimation of variance components. Input is ignored if random=NULL.

Details

The formula syntax is similar to that used in lm and many other R regression functions. Use y~x to predict the response y from the predictor x. Use y~x1+x2 to fit an additive model of the predictors x1 and x2, and use y~x1*x2 to fit an interaction model. The syntax y~x1*x2 includes the interaction and main effects, whereas the syntax y~x1:x2 is not supported. See Computational Details for specifics about how nonparametric effects are estimated.

See bigspline for definitions of type="cub", type="cub0", and type="per" splines, which can handle one-dimensional predictors. See Appendix of Helwig and Ma (2015) for information about type="tps" and type="nom" splines. Note that type="tps" can handle one-, two-, or three-dimensional predictors. I recommend using type="cub" if the predictor scores have no extreme outliers; when outliers are present, type="tps" may produce a better result.

Using the rounding parameter input rparm can greatly speed-up and stabilize the fitting for large samples. For typical cases, I recommend using rparm=0.01 for cubic and periodic splines, but smaller rounding parameters may be needed for particularly jagged functions. For thin-plate splines, the data are NOT transformed to the interval [0,1] before fitting, so the rounding parameter should be on the raw data scale. Also, for type="tps" you can enter one rounding parameter for each predictor dimension. Use rparm=1 for ordinal and nominal splines.

Value

fitted.values	Vector of fitted values corresponding to the original data points in xvars (if rparm=NA) or the rounded data points in xunique (if rparm is used).
se.fit	Vector of standard errors of fitted.values (if input se.fit=TRUE).
yvar	Response vector.
xvars	List of predictors.
type	Type of smoothing spline that was used for each predictor.

yunique	Mean of yvar for unique points after rounding (if rparm is used).
xunique	Unique rows of xvars after rounding (if rparm is used).
sigma	Estimated error standard deviation, i.e., $\hat{\sigma}$.
ndf	Data frame with two elements: n is total sample size, and df is effective degrees of freedom of fit model (trace of smoothing matrix).
info	Model fit information: vector containing the GCV, multiple R-squared, AIC, and BIC of fit model (assuming Gaussian error).
modelspec	List containing specifics of fit model (needed for prediction).
converged	Convergence status: converged=TRUE if iterative update converged, converged=FALSE if iterative update failed to converge, and converged=NA if option skip.iter=TRUE was used.
tnames	Names of the terms in model.
random	Random effects formula (same as input).
tau	Variance parameters such that sigma*sqrt(tau) gives standard deviation of random effects (if !is.null(random)).
blup	Best linear unbiased predictors (if !is.null(random)).
call	Called model in input formula.

Warnings

Cubic and cubic periodic splines transform the predictor to the interval [0,1] before fitting.

When using rounding parameters, output fitted.values corresponds to unique rounded predictor scores in output xunique. Use predict.bigssa function to get fitted values for full yvar vector.

Computational Details

To estimate η I minimize the penalized least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\eta(\mathbf{x}_i)\right)^2+\lambda J(\eta)$$

where $J(\cdot)$ is a nonnegative penalty functional quantifying the roughness of η and $\lambda > 0$ is a smoothing parameter controlling the trade-off between fitting and smoothing the data. Note that for $p > 1$ nonparametric predictors, there are additional θ_k smoothing parameters embedded in J.

The penalized least squares functioncal can be rewritten as

$$\|\mathbf{y}-\mathbf{K}\mathbf{d}-\mathbf{J}_\theta\mathbf{c}\|^2+n\lambda\mathbf{c}'\mathbf{Q}_\theta\mathbf{c}$$

where $\mathbf{K}=\{\phi(x_i)\}_{n\times m}$ is the null (parametric) space basis function matrix, $\mathbf{J}_\theta=\sum_{k=1}^{s}\theta_k\mathbf{J}_k$ with $\mathbf{J}_k=\{\rho_k(\mathbf{x}_i,\mathbf{x}_h^*)\}_{n\times q}$ denoting the k-th contrast space basis funciton matrix, $\mathbf{Q}_\theta=\sum_{k=1}^{s}\theta_k\mathbf{Q}_k$ with $\mathbf{Q}_k=\{\rho_k(\mathbf{x}_g^*,\mathbf{x}_h^*)\}_{q\times q}$ denoting the k-th penalty matrix, and $\mathbf{d}=(d_0,\ldots,d_m)'$ and $\mathbf{c}=(c_1,\ldots,c_q)'$ are the unknown basis function coefficients. The optimal smoothing parameters are chosen by minimizing the GCV score (see bigspline).

Note that this function uses the efficient SSA reparameterization described in Helwig (2013) and Helwig and Ma (2015); using is parameterization, there is one unique smoothing parameter per predictor (γ_j), and these γ_j parameters determine the structure of the θ_k parameters in the tensor product space. To evaluate the GCV score, this function uses the improved (scalable) SSA algorithm discussed in Helwig (2013) and Helwig and Ma (2015).

Skip Iteration

For $p > 1$ predictors, initial values for the γ_j parameters (that determine the structure of the θ_k parameters) are estimated using the smart starting algorithm described in Helwig (2013) and Helwig and Ma (2015).

Default use of this function (`skip.iter=TRUE`) fixes the γ_j parameters afer the smart start, and then finds the global smoothing parameter λ (among the input `lambdas`) that minimizes the GCV score. This approach typically produces a solution very similar to the more optimal solution using `skip.iter=FALSE`.

Setting `skip.iter=FALSE` uses the same smart starting algorithm as setting `skip.iter=TRUE`. However, instead of fixing the γ_j parameters afer the smart start, using `skip.iter=FALSE` iterates between estimating the optimal λ and the optimal γ_j parameters. The R function `nlm` is used to minimize the GCV score with respect to the γ_j parameters, which can be time consuming for models with many predictors and/or a large number of knots.

Random Effects

The input `random` adds random effects to the model assuming a variance components structure. Both nested and crossed random effects are supported. In all cases, the random effects are assumed to be indepedent zero-mean Gaussian variables with the variance depending on group membership.

Random effects are distinguished by vertical bars ("|"), which separate expressions for design matrices (left) from group factors (right). For example, the syntax `~1|group` includes a random intercept for each level of `group`, whereas the syntax `~1+x|group` includes both a random intercept and a random slope for each level of `group`. For crossed random effects, parentheses are needed to distinguish different terms, e.g., `~(1|group1)+(1|group2)` includes a random intercept for each level of `group1` and a random intercept for each level of `group2`, where both `group1` and `group2` are factors. For nested random effects, the syntax `~group|subject` can be used, where both `group` and `subject` are factors such that the levels of `subject` are nested within those of `group`.

The input `remlalg` determines the REML algorithm used to estimate the variance components. Setting `remlalg="FS"` uses a Fisher Scoring algorithm (default). Setting `remlalg="NR"` uses a Newton-Raphson algorithm. Setting `remlalg="EM"` uses an Expectation Maximization algorithm. Use `remlalg="none"` to fit a model with known variance components (entered through `remltau`).

The input `remliter` sets the maximum number of iterations for the REML estimation. The input `remltol` sets the convergence tolerance for the REML estimation, which is determined via relative change in the REML log-likelihood. The input `remltau` sets the initial estimates of variance parameters; default is `remltau = rep(1,ntau)` where `ntau` is the number of variance components.

Note

The spline is estimated using penalized least-squares, which does not require the Gaussian error assumption. However, the spline inference information (e.g., standard errors and fit information) requires the Gaussian error assumption.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define univariate function and data
set.seed(773)
myfun <- function(x){ sin(2*pi*x) }
x <- runif(500)
y <- myfun(x) + rnorm(500)

# cubic, periodic, and thin-plate spline models with 20 knots
cubmod <- bigssa(y~x,type="cub",nknots=20,se.fit=TRUE)
cubmod
permod <- bigssa(y~x,type="per",nknots=20,se.fit=TRUE)
permod
tpsmod <- bigssa(y~x,type="tps",nknots=20,se.fit=TRUE)
tpsmod


##########   EXAMPLE 2   ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){sin(2*pi*x1v)+log(x2v+.1)+cos(pi*(x1v-x2v))}
x1v <- runif(500)
x2v <- runif(500)
y <- myfun(x1v,x2v) + rnorm(500)

# cubic splines with 50 randomly selected knots
intmod <- bigssa(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
intmod
crossprod( myfun(x1v,x2v) - intmod$fitted.values )/500

# fit additive model (with same knots)
addmod <- bigssa(y~x1v+x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
addmod
```

```
crossprod( myfun(x1v,x2v) - addmod$fitted.values )/500


##########   EXAMPLE 3   ##########

# function with two continuous and one nominal predictor (3 levels)
set.seed(773)
myfun <- function(x1v,x2v,x3v){
  fval <- rep(0,length(x1v))
  xmeans <- c(-1,0,1)
  for(j in 1:3){
    idx <- which(x3v==letters[j])
    fval[idx] <- xmeans[j]
  }
  fval[idx] <- fval[idx] + cos(4*pi*(x1v[idx]))
  fval <- (fval + sin(3*pi*x1v*x2v+pi)) / sqrt(2)
}
x1v <- runif(500)
x2v <- runif(500)
x3v <- sample(letters[1:3],500,replace=TRUE)
y <- myfun(x1v,x2v,x3v) + rnorm(500)

# 3-way interaction with 50 knots
cuimod <- bigssa(y~x1v*x2v*x3v,type=list(x1v="cub",x2v="cub",x3v="nom"),nknots=50)
crossprod( myfun(x1v,x2v,x3v) - cuimod$fitted.values )/500

# fit correct interaction model with 50 knots
cubmod <- bigssa(y~x1v*x2v+x1v*x3v,type=list(x1v="cub",x2v="cub",x3v="nom"),nknots=50)
crossprod( myfun(x1v,x2v,x3v) - cubmod$fitted.values )/500

# fit model using 2-dimensional thin-plate and nominal
x1new <- cbind(x1v,x2v)
x2new <- x3v
tpsmod <- bigssa(y~x1new*x2new,type=list(x1new="tps",x2new="nom"),nknots=50)
crossprod( myfun(x1v,x2v,x3v) - tpsmod$fitted.values )/500


##########   EXAMPLE 4   ##########

# function with four continuous predictors
set.seed(773)
myfun <- function(x1v,x2v,x3v,x4v){
  sin(2*pi*x1v) + log(x2v+.1) + x3v*cos(pi*(x4v))
  }
x1v <- runif(500)
x2v <- runif(500)
x3v <- runif(500)
x4v <- runif(500)
y <- myfun(x1v,x2v,x3v,x4v) + rnorm(500)

# fit cubic spline model with x3v*x4v interaction
cubmod <- bigssa(y~x1v+x2v+x3v*x4v,type=list(x1v="cub",x2v="cub",x3v="cub",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500
```

bigssg *Fits Generalized Smoothing Spline ANOVA Models*

Description

Given an exponential family response vector $\mathbf{y} = \{y_i\}_{n\times 1}$, a Generalized Smoothing Spline Anova (GSSA) has the form

$$g(\mu_i) = \eta(\mathbf{x}_i)$$

where μ_i is the expected value of the i-th observation's respone, $g(\cdot)$ is some invertible link function, $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ is the i-th observation's nonparametric predictor vector, and η is an unknown smooth function relating the response and nonparametric predictors. Function can fit additive models, and also allows for 2-way and 3-way interactions between any number of predictors. Response can be one of five non-Gaussian distributions: Binomial, Poisson, Gamma, Inverse Gaussian, or Negative Binomial (see Details and Examples).

Usage

```
bigssg(formula,family,data=NULL,type=NULL,nknots=NULL,rparm=NA,
       lambdas=NULL,skip.iter=TRUE,se.lp=FALSE,rseed=1234,
       gcvopts=NULL,knotcheck=TRUE,gammas=NULL,weights=NULL,
       gcvtype=c("acv","gacv","gacv.old"))
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted (see Details and Examples for more information).
family	Distribution for response. One of five options: "binomial", "poisson", "Gamma", "inverse.gaussian", or "negbin". See Response section.
data	Optional data frame, list, or environment containing the variables in formula. Or an object of class "makessg", which is output from makessg.
type	List of smoothing spline types for predictors in formula (see Details). Options include type="cub" for cubic, type="cub0" for another cubic, type="per" for cubic periodic, type="tps" for cubic thin-plate, type="ord" for ordinal, and type="nom" for nominal.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of data to use as knots.
rparm	List of rounding parameters for each predictor. See Details.
lambdas	Vector of global smoothing parameters to try. Default lambdas=10^-c(9:0).
skip.iter	Logical indicating whether to skip the iterative smoothing parameter update. Using skip.iter=FALSE should provide a more optimal solution, but the fitting time may be substantially longer. See Skip Iteration section.
se.lp	Logical indicating if the standard errors of the linear predictors (η) should be estimated.

rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.
gcvopts	Control parameters for optimization. List with 6 elements: (i) maxit: maximum number of outer iterations, (ii) gcvtol: covergence tolerance for iterative GACV update, (iii) alpha: tuning parameter for GACV minimization, (iv) inmaxit: maximum number of inner iterations for iteratively reweighted fitting, (v) intol: inner convergence tolerance for iteratively reweighted fitting, and (vi) insub: number of data points to subsample when checking inner convergence. gcvopts=list(maxit=5,gcvtol=10^-5,alpha=1,inmaxit=100,intol=10^-5,insub=10^4)
knotcheck	If TRUE, only unique knots are used (for stability).
gammas	List of initial smoothing parameters for each predictor. See Details.
weights	Vector of positive weights for fitting (default is vector of ones).
gcvtype	Cross-validation criterion for selecting smoothing parameters (see Details).

Details

The formula syntax is similar to that used in lm and many other R regression functions. Use y~x to predict the response y from the predictor x. Use y~x1+x2 to fit an additive model of the predictors x1 and x2, and use y~x1*x2 to fit an interaction model. The syntax y~x1*x2 includes the interaction and main effects, whereas the syntax y~x1:x2 is not supported. See Computational Details for specifics about how nonparametric effects are estimated.

See bigspline for definitions of type="cub", type="cub0", and type="per" splines, which can handle one-dimensional predictors. See Appendix of Helwig and Ma (2015) for information about type="tps" and type="nom" splines. Note that type="tps" can handle one-, two-, or three-dimensional predictors. I recommend using type="cub" if the predictor scores have no extreme outliers; when outliers are present, type="tps" may produce a better result.

Using the rounding parameter input rparm can greatly speed-up and stabilize the fitting for large samples. For typical cases, I recommend using rparm=0.01 for cubic and periodic splines, but smaller rounding parameters may be needed for particularly jagged functions. For thin-plate splines, the data are NOT transformed to the interval [0,1] before fitting, so rounding parameter should be on raw data scale. Also, for type="tps" you can enter one rounding parameter for each predictor dimension. Use rparm=1 for ordinal and nominal splines.

Value

fitted.values	Vector of fitted values (data scale) corresponding to the original data points in xvars (if rparm=NA) or the rounded data points in xunique (if rparm is used).
linear.predictors	Vector of fitted values (link scale) corresponding to the original data points in xvars (if rparm=NA) or the rounded data points in xunique (if rparm is used).
se.lp	Vector of standard errors of linear.predictors (if input se.lp=TRUE).
yvar	Response vector.
xvars	List of predictors.
type	Type of smoothing spline that was used for each predictor.

yunique	Mean of yvar for unique points after rounding (if rparm is used).
xunique	Unique rows of xvars after rounding (if rparm is used).
dispersion	Estimated dispersion parameter (see Response section).
ndf	Data frame with two elements: n is total sample size, and df is effective degrees of freedom of fit model (trace of smoothing matrix).
info	Model fit information: vector containing the GCV, multiple R-squared, AIC, and BIC of fit model.
modelspec	List containing specifics of fit model (needed for prediction).
converged	Convergence status: converged=TRUE if iterative update converged, converged=FALSE if iterative update failed to converge, and converged=NA if option skip.iter=TRUE was used.
tnames	Names of the terms in model.
family	Distribution family (same as input).
call	Called model in input formula.

Warnings

Cubic and cubic periodic splines transform the predictor to the interval [0,1] before fitting.

When using rounding parameters, output fitted.values corresponds to unique rounded predictor scores in output xunique. Use predict.bigssg function to get fitted values for full yvar vector.

Response

Only one link is permitted for each family:

family="binomial" Logit link. Response should be vector of proportions in the interval [0,1]. If response is a sample proportion, the total count should be input through weights argument.

family="poisson" Log link. Response should be vector of counts (non-negative integers).

family="Gamma" Inverse link. Response should be vector of positive real-valued data. Estimated dispersion parameter is the inverse of the shape parameter, so that the variance of the response increases as dispersion increases.

family="inverse.gaussian" Inverse-square link. Response should be vector of positive real-valued data. Estimated dispersion parameter is the inverse of the shape parameter, so that the variance of the response increases as dispersion increases.

family="negbin" Log link. Response should be vector of counts (non-negative integers). Estimated dispersion parameter is the inverse of the size parameter, so that the variance of the response increases as dispersion increases.

family=list("negbin",2) Log link. Response should be vector of counts (non-negative integers). Second element is the known (common) dispersion parameter (2 in this case). The input dispersion parameter should be the inverse of the size parameter, so that the variance of the response increases as dispersion increases.

Computational Details

To estimate η I minimize the (negative of the) penalized log likelihood

$$-\frac{1}{n}\sum_{i=1}^{n}\{y_i\eta(\mathbf{x}_i)-b(\eta(\mathbf{x}_i))\}+\frac{\lambda}{2}J(\eta)$$

where $J(\cdot)$ is a nonnegative penalty functional quantifying the roughness of η and $\lambda > 0$ is a smoothing parameter controlling the trade-off between fitting and smoothing the data. Note that for $p > 1$ nonparametric predictors, there are additional θ_k smoothing parameters embedded in J.

Following standard exponential family theory, $\mu_i = \dot{b}(\eta(\mathbf{x}_i))$ and $v_i = \ddot{b}(\eta(\mathbf{x}_i))a(\xi)$, where $\dot{b}(\cdot)$ and $\ddot{b}(\cdot)$ denote the first and second derivatives of $b(\cdot)$, v_i is the variance of y_i,and ξ is the dispersion parameter. Given fixed smoothing parameters, the optimal η can be estimated by iteratively minimizing the penalized reweighted least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}v_i^*\left(y_i^*-\eta(\mathbf{x}_i)\right)^2+\lambda J(\eta)$$

where $v_i^* = v_i/a(\xi)$ is the weight, $y_i^* = \hat{\eta}(\mathbf{x}_i) + (y_i - \hat{\mu}_i)/v_i^*$ is the adjusted dependent variable, and $\hat{\eta}(\mathbf{x}_i)$ is the current estimate of η.

The optimal smoothing parameters are chosen via direct cross-validation (see Gu & Xiang, 2001).

Setting `gcvtype="acv"` uses the Approximate Cross-Validation (ACV) score:

$$-\frac{1}{n}\sum_{i=1}^{n}\{y_i\hat{\eta}(\mathbf{x}_i)-b(\hat{\eta}(\mathbf{x}_i))\}+\frac{1}{n}\sum_{i=1}^{n}\frac{s_{ii}}{(1-s_{ii})v_i^*}y_i(y_i-\hat{\mu}_i)$$

where s_{ii} is the i-th diagonal of the smoothing matrix $\mathbf{S}_{\boldsymbol{\lambda}}$.

Setting `gcvtype="gacv"` uses the Generalized ACV (GACV) score:

$$-\frac{1}{n}\sum_{i=1}^{n}\{y_i\hat{\eta}(\mathbf{x}_i)-b(\hat{\eta}(\mathbf{x}_i))\}+\frac{\mathrm{tr}(\mathbf{S}_{\boldsymbol{\lambda}}\mathbf{V}^{-1})}{n-\mathrm{tr}(\mathbf{S}_{\boldsymbol{\lambda}})}\frac{1}{n}\sum_{i=1}^{n}y_i(y_i-\hat{\mu}_i)$$

where $\mathbf{S}_{\boldsymbol{\lambda}}$ is the smoothing matrix, and $\mathbf{V} = \mathrm{diag}(v_1^*, \ldots, v_n^*)$.

Setting `gcvtype="gacv.old"` uses an approximation of the GACV where $\frac{1}{n}\mathrm{tr}(\mathbf{S}_{\boldsymbol{\lambda}}\mathbf{V}^{-1})$ is approximated using $\frac{1}{n^2}\mathrm{tr}(\mathbf{S}_{\boldsymbol{\lambda}})\mathrm{tr}(\mathbf{V}^{-1})$. This option is included for back-compatibility (ver 1.0-4 and earlier), and is not recommended because the ACV or GACV often perform better.

Note that this function uses the efficient SSA reparameterization described in Helwig (2013) and Helwig and Ma (2015); using is parameterization, there is one unique smoothing parameter per predictor (γ_j), and these γ_j parameters determine the structure of the θ_k parameters in the tensor product space. To evaluate the ACV/GACV score, this function uses the improved (scalable) GSSA algorithm discussed in Helwig (in preparation).

Skip Iteration

For $p > 1$ predictors, initial values for the γ_j parameters (that determine the structure of the θ_k parameters) are estimated using an extension of the smart starting algorithm described in Helwig (2013) and Helwig and Ma (2015).

Default use of this function (`skip.iter=TRUE`) fixes the γ_j parameters afer the smart start, and then finds the global smoothing parameter λ (among the input `lambdas`) that minimizes the GCV score. This approach typically produces a solution very similar to the more optimal solution using `skip.iter=FALSE`.

Setting `skip.iter=FALSE` uses the same smart starting algorithm as setting `skip.iter=TRUE`. However, instead of fixing the γ_j parameters afer the smart start, using `skip.iter=FALSE` iterates between estimating the optimal λ and the optimal γ_j parameters. The R function `nlm` is used to minimize the approximate GACV score with respect to the γ_j parameters, which can be time consuming for models with many predictors and/or a large number of knots.

Note

The spline is estimated using penalized likelihood estimation. Standard errors of the linear predictors are formed using Bayesian confidence intervals.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Gu, C. and Xiang, D. (2001). Cross-validating non-Gaussian data: Generalized approximate cross-validation revisited. *Journal of Computational and Graphical Statistics, 10*, 581-591.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1 (1-way GSSA)   ##########

# define univariate function and data
set.seed(1)
myfun <- function(x){ sin(2*pi*x) }
ndpts <- 1000
x <- runif(ndpts)

# binomial response (no weights)
set.seed(773)
lp <- myfun(x)
p <- 1/(1+exp(-lp))
y <- rbinom(n=ndpts,size=1,p=p)     ## y is binary data
gmod <- bigssg(y~x,family="binomial",type="cub",nknots=20)
crossprod( lp - gmod$linear.predictor )/length(lp)

# binomial response (with weights)
```

```
set.seed(773)
lp <- myfun(x)
p <- 1/(1+exp(-lp))
w <- sample(c(10,20,30,40,50),length(p),replace=TRUE)
y <- rbinom(n=ndpts,size=w,p=p)/w   ## y is proportion correct
gmod <- bigssg(y~x,family="binomial",type="cub",nknots=20,weights=w)
crossprod( lp - gmod$linear.predictor )/length(lp)

# poisson response
set.seed(773)
lp <- myfun(x)
mu <- exp(lp)
y <- rpois(n=ndpts,lambda=mu)
gmod <- bigssg(y~x,family="poisson",type="cub",nknots=20)
crossprod( lp - gmod$linear.predictor )/length(lp)

# Gamma response
set.seed(773)
lp <- myfun(x) + 2
mu <- 1/lp
y <- rgamma(n=ndpts,shape=4,scale=mu/4)
gmod <- bigssg(y~x,family="Gamma",type="cub",nknots=20)
1/gmod$dispersion   ## dispersion = 1/shape
crossprod( lp - gmod$linear.predictor )/length(lp)

# inverse gaussian response (not run: requires statmod package)
# require(statmod)
# set.seed(773)
# lp <- myfun(x) + 2
# mu <- sqrt(1/lp)
# y <- rinvgauss(n=ndpts,mean=mu,shape=2)
# gmod <- bigssg(y~x,family="inverse.gaussian",type="cub",nknots=20)
# 1/gmod$dispersion   ## dispersion = 1/shape
# crossprod( lp - gmod$linear.predictor )/length(lp)

# negative binomial response (known dispersion)
set.seed(773)
lp <- myfun(x)
mu <- exp(lp)
y <- rnbinom(n=ndpts,size=.5,mu=mu)
gmod <- bigssg(y~x,family=list("negbin",2),type="cub",nknots=20)
1/gmod$dispersion   ## dispersion = 1/size
crossprod( lp - gmod$linear.predictor )/length(lp)

# negative binomial response (unknown dispersion)
set.seed(773)
lp <- myfun(x)
mu <- exp(lp)
y <- rnbinom(n=ndpts,size=.5,mu=mu)
gmod <- bigssg(y~x,family="negbin",type="cub",nknots=20)
1/gmod$dispersion   ## dispersion = 1/size
crossprod( lp - gmod$linear.predictor )/length(lp)
```

```
## Not run:

##########   EXAMPLE 2 (2-way GSSA)   ##########

# function with two continuous predictors
set.seed(1)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
ndpts <- 1000
x1v <- runif(ndpts)
x2v <- runif(ndpts)

# binomial response (no weights)
set.seed(773)
lp <- myfun(x1v,x2v)
p <- 1/(1+exp(-lp))
y <- rbinom(n=ndpts,size=1,p=p)     ## y is binary data
gmod <- bigssg(y~x1v*x2v,family="binomial",type=list(x1v="cub",x2v="cub"),nknots=50)
crossprod( lp - gmod$linear.predictor )/length(lp)

# binomial response (with weights)
set.seed(773)
lp <- myfun(x1v,x2v)
p <- 1/(1+exp(-lp))
w <- sample(c(10,20,30,40,50),length(p),replace=TRUE)
y <- rbinom(n=ndpts,size=w,p=p)/w   ## y is proportion correct
gmod <- bigssg(y~x1v*x2v,family="binomial",type=list(x1v="cub",x2v="cub"),nknots=50,weights=w)
crossprod( lp - gmod$linear.predictor )/length(lp)

# poisson response
set.seed(773)
lp <- myfun(x1v,x2v)
mu <- exp(lp)
y <- rpois(n=ndpts,lambda=mu)
gmod <- bigssg(y~x1v*x2v,family="poisson",type=list(x1v="cub",x2v="cub"),nknots=50)
crossprod( lp - gmod$linear.predictor )/length(lp)

# Gamma response
set.seed(773)
lp <- myfun(x1v,x2v)+6
mu <- 1/lp
y <- rgamma(n=ndpts,shape=4,scale=mu/4)
gmod <- bigssg(y~x1v*x2v,family="Gamma",type=list(x1v="cub",x2v="cub"),nknots=50)
1/gmod$dispersion   ## dispersion = 1/shape
crossprod( lp - gmod$linear.predictor )/length(lp)

# inverse gaussian response (not run: requires 'statmod' package)
# require(statmod)
# set.seed(773)
# lp <- myfun(x1v,x2v)+6
# mu <- sqrt(1/lp)
# y <- rinvgauss(n=ndpts,mean=mu,shape=2)
```

```
# gmod <- bigssg(y~x1v*x2v,family="inverse.gaussian",type=list(x1v="cub",x2v="cub"),nknots=50)
# 1/gmod$dispersion   ## dispersion = 1/shape
# crossprod( lp - gmod$linear.predictor )/length(lp)

# negative binomial response (known dispersion)
set.seed(773)
lp <- myfun(x1v,x2v)
mu <- exp(lp)
y <- rnbinom(n=ndpts,size=.5,mu=mu)
gmod <- bigssg(y~x1v*x2v,family=list("negbin",2),type=list(x1v="cub",x2v="cub"),nknots=50)
1/gmod$dispersion   ## dispersion = 1/size
crossprod( lp - gmod$linear.predictor )/length(lp)

# negative binomial response (unknown dispersion)
set.seed(773)
lp <- myfun(x1v,x2v)
mu <- exp(lp)
y <- rnbinom(n=ndpts,size=.5,mu=mu)
gmod <- bigssg(y~x1v*x2v,family="negbin",type=list(x1v="cub",x2v="cub"),nknots=50)
1/gmod$dispersion   ## dispersion = 1/size
crossprod( lp - gmod$linear.predictor )/length(lp)

## End(Not run)
```

---

bigssp *Fits Smoothing Splines with Parametric Effects*

Description

Given a real-valued response vector $\mathbf{y} = \{y_i\}_{n\times 1}$, a semiparametric regression model has the form

$$y_i = \eta(\mathbf{x}_i) + \sum_{j=1}^{t} b_j z_{ij} + e_i$$

where y_i is the i-th observation's respone, $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ is the i-th observation's nonparametric predictor vector, η is an unknown smooth function relating the response and nonparametric predictors, $\mathbf{z}_i = (z_{i1}, \ldots, z_{it})$ is the i-th observation's parametric predictor vector, and $e_i \sim \mathrm{N}(0, \sigma^2)$ is iid Gaussian error. Function can fit both additive and interactive non/parametric effects, and allows for 2-way and 3-way interactions between nonparametric and parametric effects (see Details and Examples).

Usage

```
bigssp(formula,data=NULL,type=NULL,nknots=NULL,rparm=NA,
       lambdas=NULL,skip.iter=TRUE,se.fit=FALSE,rseed=1234,
       gcvopts=NULL,knotcheck=TRUE,thetas=NULL,weights=NULL,
       random=NULL,remlalg=c("FS","NR","EM","none"),remliter=500,
       remltol=10^-4,remltau=NULL)
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted (see Details and Examples for more information).
data	Optional data frame, list, or environment containing the variables in formula. Or an object of class "makessp", which is output from makessp.
type	List of smoothing spline types for predictors in formula (see Details). Options include type="cub" for cubic, type="cub0" for another cubic, type="per" for cubic periodic, type="tps" for cubic thin-plate, type="ord" for ordinal, and type="nom" for nominal. Use type="prm" for parametric effect.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of data to use as knots.
rparm	List of rounding parameters for each predictor. See Details.
lambdas	Vector of global smoothing parameters to try. Default lambdas=10^-c(9:0).
skip.iter	Logical indicating whether to skip the iterative smoothing parameter update. Using skip.iter=FALSE should provide a more optimal solution, but the fitting time may be substantially longer. See Skip Iteration section.
se.fit	Logical indicating if the standard errors of the fitted values should be estimated.
rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.
gcvopts	Control parameters for optimization. List with 3 elements: (a) maxit: maximum number of algorithm iterations, (b) gcvtol: covergence tolerance for iterative GCV update, and (c) alpha: tuning parameter for GCV minimization. Default: gcvopts=list(maxit=5,gcvtol=10^-5,alpha=1)
knotcheck	If TRUE, only unique knots are used (for stability).
thetas	List of initial smoothing parameters for each predictor subspace. See Details.
weights	Vector of positive weights for fitting (default is vector of ones).
random	Adds random effects to model (see Random Effects section).
remlalg	REML algorithm for estimating variance components (see Random Effects section). Input is ignored if random=NULL.
remliter	Maximum number of iterations for REML estimation of variance components. Input is ignored if random=NULL.
remltol	Convergence tolerance for REML estimation of variance components. Input is ignored if random=NULL.
remltau	Initial estimate of variance parameters for REML estimation of variance components. Input is ignored if random=NULL.

Details

The formula syntax is similar to that used in lm and many other R regression functions. Use y~x to predict the response y from the predictor x. Use y~x1+x2 to fit an additive model of the predictors x1 and x2, and use y~x1*x2 to fit an interaction model. The syntax y~x1*x2 includes the interaction

and main effects, whereas the syntax `y~x1:x2` only includes the interaction. See Computational Details for specifics about how non/parametric effects are estimated.

See `bigspline` for definitions of `type="cub"`, `type="cub0"`, and `type="per"` splines, which can handle one-dimensional predictors. See Appendix of Helwig and Ma (2015) for information about `type="tps"` and `type="nom"` splines. Note that `type="tps"` can handle one-, two-, or three-dimensional predictors. I recommend using `type="cub"` if the predictor scores have no extreme outliers; when outliers are present, `type="tps"` may produce a better result.

Using the rounding parameter input `rparm` can greatly speed-up and stabilize the fitting for large samples. For typical cases, I recommend using `rparm=0.01` for cubic and periodic splines, but smaller rounding parameters may be needed for particularly jagged functions. For thin-plate splines, the data are NOT transformed to the interval [0,1] before fitting, so the rounding parameter should be on the raw data scale. Also, for `type="tps"` you can enter one rounding parameter for each predictor dimension. Use `rparm=1` for ordinal and nominal splines.

Value

`fitted.values`	Vector of fitted values corresponding to the original data points in `xvars` (if `rparm=NA`) or the rounded data points in `xunique` (if `rparm` is used).
`se.fit`	Vector of standard errors of `fitted.values` (if input `se.fit=TRUE`).
`yvar`	Response vector.
`xvars`	List of predictors.
`type`	Type of smoothing spline that was used for each predictor.
`yunique`	Mean of `yvar` for unique points after rounding (if `rparm` is used).
`xunique`	Unique rows of `xvars` after rounding (if `rparm` is used).
`sigma`	Estimated error standard deviation, i.e., $\hat{\sigma}$.
`ndf`	Data frame with two elements: `n` is total sample size, and `df` is effective degrees of freedom of fit model (trace of smoothing matrix).
`info`	Model fit information: vector containing the GCV, multiple R-squared, AIC, and BIC of fit model (assuming Gaussian error).
`modelspec`	List containing specifics of fit model (needed for prediction).
`converged`	Convergence status: `converged=TRUE` if iterative update converged, `converged=FALSE` if iterative update failed to converge, and `converged=NA` if option `skip.iter=TRUE` was used.
`tnames`	Names of the terms in model.
`random`	Random effects formula (same as input).
`tau`	Variance parameters such that `sigma*sqrt(tau)` gives standard deviation of random effects (if `!is.null(random)`).
`blup`	Best linear unbiased predictors (if `!is.null(random)`).
`call`	Called model in input `formula`.

Warnings

Cubic and cubic periodic splines transform the predictor to the interval [0,1] before fitting.

When using rounding parameters, output `fitted.values` corresponds to unique rounded predictor scores in output `xunique`. Use `predict.bigssp` function to get fitted values for full `yvar` vector.

Computational Details

To estimate η I minimize the penalized least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \eta(\mathbf{x}_i) - \sum_{j=1}^{t} b_j z_{ij}\right)^2 + \lambda J(\eta)$$

where $J(\cdot)$ is a nonnegative penalty functional quantifying the roughness of η and $\lambda > 0$ is a smoothing parameter controlling the trade-off between fitting and smoothing the data. Note that for $p > 1$ nonparametric predictors, there are additional θ_k smoothing parameters embedded in J.

The penalized least squares functioncal can be rewritten as

$$\|\mathbf{y} - \mathbf{K}\mathbf{d} - \mathbf{J}_\theta\mathbf{c}\|^2 + n\lambda\mathbf{c}'\mathbf{Q}_\theta\mathbf{c}$$

where $\mathbf{K} = \{\phi(x_i), \mathbf{z}_i\}_{n\times m}$ is the parametric space basis function matrix, $\mathbf{J}_\theta = \sum_{k=1}^{s}\theta_k\mathbf{J}_k$ with $\mathbf{J}_k = \{\rho_k(\mathbf{x}_i, \mathbf{x}_h^*)\}_{n\times q}$ denoting the k-th contrast space basis funciton matrix, $\mathbf{Q}_\theta = \sum_{k=1}^{s}\theta_k\mathbf{Q}_k$ with $\mathbf{Q}_k = \{\rho_k(\mathbf{x}_g^*, \mathbf{x}_h^*)\}_{q\times q}$ denoting the k-th penalty matrix, and $\mathbf{d} = (d_0, \ldots, d_m)'$ and $\mathbf{c} = (c_1, \ldots, c_q)'$ are the unknown basis function coefficients. The optimal smoothing parameters are chosen by minimizing the GCV score (see `bigspline`).

Note that this function uses the classic smoothing spline parameterization (see Gu, 2013), so there is more than one smoothing parameter per predictor (if interactions are included in the model). To evaluate the GCV score, this function uses the improved (scalable) SSA algorithm discussed in Helwig (2013) and Helwig and Ma (2015).

Skip Iteration

For $p > 1$ predictors, initial values for the θ_k parameters are estimated using Algorithm 3.2 described in Gu and Wahba (1991).

Default use of this function (`skip.iter=TRUE`) fixes the θ_k parameters afer the smart start, and then finds the global smoothing parameter λ (among the input `lambdas`) that minimizes the GCV score. This approach typically produces a solution very similar to the more optimal solution using `skip.iter=FALSE`.

Setting `skip.iter=FALSE` uses the same smart starting algorithm as setting `skip.iter=TRUE`. However, instead of fixing the θ_k parameters afer the smart start, using `skip.iter=FALSE` iterates between estimating the optimal λ and the optimal θ_k parameters. The R function `nlm` is used to minimize the GCV score with respect to the θ_k parameters, which can be time consuming for models with many predictors.

Random Effects

The input `random` adds random effects to the model assuming a variance components structure. Both nested and crossed random effects are supported. In all cases, the random effects are assumed to be indepedent zero-mean Gaussian variables with the variance depending on group membership.

Random effects are distinguished by vertical bars ("|"), which separate expressions for design matrices (left) from group factors (right). For example, the syntax `~1|group` includes a random intercept for each level of `group`, whereas the syntax `~1+x|group` includes both a random intercept and a random slope for each level of `group`. For crossed random effects, parentheses are needed to distinguish different terms, e.g., `~(1|group1)+(1|group2)` includes a random intercept for each level

of group1 and a random intercept for each level of group2, where both group1 and group2 are factors. For nested random effects, the syntax ~group|subject can be used, where both group and subject are factors such that the levels of subject are nested within those of group.

The input remlalg determines the REML algorithm used to estimate the variance components. Setting remlalg="FS" uses a Fisher Scoring algorithm (default). Setting remlalg="NR" uses a Newton-Raphson algorithm. Setting remlalg="EM" uses an Expectation Maximization algorithm. Use remlalg="none" to fit a model with known variance components (entered through remltau).

The input remliter sets the maximum number of iterations for the REML estimation. The input remltol sets the convergence tolerance for the REML estimation, which is determined via relative change in the REML log-likelihood. The input remltau sets the initial estimates of variance parameters; default is remltau = rep(1,ntau) where ntau is the number of variance components.

Note

The spline is estimated using penalized least-squares, which does not require the Gaussian error assumption. However, the spline inference information (e.g., standard errors and fit information) requires the Gaussian error assumption.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

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*, 383-398.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE   ##########

# function with four continuous predictors
set.seed(773)
myfun <- function(x1v,x2v,x3v,x4v){
  sin(2*pi*x1v) + log(x2v+.1) + x3v*cos(pi*(x4v))
  }
```

```
x1v <- runif(500)
x2v <- runif(500)
x3v <- runif(500)
x4v <- runif(500)
y <- myfun(x1v,x2v,x3v,x4v) + rnorm(500)

# fit cubic spline model with x3v*x4v interaction and x3v as "cub"
# (includes x3v and x4v main effects)
cubmod <- bigssp(y~x1v+x2v+x3v*x4v,type=list(x1v="cub",x2v="cub",x3v="cub",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500

# fit cubic spline model with x3v*x4v interaction and x3v as "cub0"
# (includes x3v and x4v main effects)
cubmod <- bigssp(y~x1v+x2v+x3v*x4v,type=list(x1v="cub",x2v="cub",x3v="cub0",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500

# fit model with x3v*x4v interaction treating x3v as parametric effect
# (includes x3v and x4v main effects)
cubmod <- bigssp(y~x1v+x2v+x3v*x4v,type=list(x1v="cub",x2v="cub",x3v="prm",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500

# fit cubic spline model with x3v:x4v interaction and x3v as "cub"
# (excludes x3v and x4v main effects)
cubmod <- bigssp(y~x1v+x2v+x3v:x4v,type=list(x1v="cub",x2v="cub",x3v="cub",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500

# fit cubic spline model with x3v:x4v interaction and x3v as "cub0"
# (excludes x3v and x4v main effects)
cubmod <- bigssp(y~x1v+x2v+x3v:x4v,type=list(x1v="cub",x2v="cub",x3v="cub0",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500

# fit model with x3v:x4v interaction treating x3v as parametric effect
# (excludes x3v and x4v main effects)
cubmod <- bigssp(y~x1v+x2v+x3v:x4v,type=list(x1v="cub",x2v="cub",x3v="prm",x4v="cub"),nknots=50)
crossprod( myfun(x1v,x2v,x3v,x4v) - cubmod$fitted.values )/500
```

---

bigtps *Fits Cubic Thin-Plate Splines*

Description

Given a real-valued response vector $\mathbf{y} = \{y_i\}_{n\times 1}$, a thin-plate spline model has the form

$$y_i = \eta(\mathbf{x}_i) + e_i$$

where y_i is the i-th observation's respone, $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$ is the i-th observation's nonparametric predictor vector, η is an unknown smooth function relating the response and predictor, and $e_i \sim \mathrm{N}(0, \sigma^2)$ is iid Gaussian error. Function only fits interaction models.

Usage

```
bigtps(x,y,nknots=NULL,nvec=NULL,rparm=NA,
       alpha=1,lambdas=NULL,se.fit=FALSE,
       rseed=1234,knotcheck=TRUE)
```

Arguments

x	Predictor vector or matrix with three or less columns.
y	Response vector. Must be same length as x has rows.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of x to use as knots.
nvec	Number of eigenvectors (and eigenvalues) to use in approximation. Must be less than or equal to the number of knots and greater than or equal to ncol(x)+2. Default sets nvec<-nknots. Can also input 0<nvec<1 to retain nvec percentage of eigenbasis variation.
rparm	Rounding parameter(s) for x. Use rparm=NA to fit unrounded solution. Can provide one (positive) rounding parameter for each column of x.
alpha	Manual tuning parameter for GCV score. Using alpha=1 gives unbaised esitmate. Using a larger alpha enforces a smoother estimate.
lambdas	Vector of global smoothing parameters to try. Default estimates smoothing parameter that minimizes GCV score.
se.fit	Logical indicating if the standard errors of fitted values should be estimated.
rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.
knotcheck	If TRUE, only unique knots are used (for stability).

Details

To estimate η I minimize the penalized least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \eta(\mathbf{x}_i))^2 + \lambda J(\eta)$$

where $J(\eta)$ is the thin-plate penalty (see Helwig and Ma) and $\lambda \geq 0$ is a smoothing parameter that controls the trade-off between fitting and smoothing the data. Default use of the function estimates λ by minimizing the GCV score (see bigspline).

Using the rounding parameter input rparm can greatly speed-up and stabilize the fitting for large samples. When rparm is used, the spline is fit to a set of unique data points after rounding; the unique points are determined using the efficient algorithm described in Helwig (2013). Rounding parameter should be on the raw data scale.

Value

`fitted.values`	Vector of fitted values corresponding to the original data points in `x` (if `rparm=NA`) or the rounded data points in `xunique` (if `rparm` is used).
`se.fit`	Vector of standard errors of `fitted.values` (if input `se.fit=TRUE`).
`x`	Predictor vector (same as input).
`y`	Response vector (same as input).
`xunique`	Unique elements of `x` after rounding (if `rparm` is used).
`yunique`	Mean of `y` for unique elements of `x` after rounding (if `rparm` is used).
`funique`	Vector giving frequency of each element of `xunique` (if `rparm` is used).
`sigma`	Estimated error standard deviation, i.e., $\hat{\sigma}$.
`ndf`	Data frame with two elements: `n` is total sample size, and `df` is effective degrees of freedom of fit model (trace of smoothing matrix).
`info`	Model fit information: vector containing the GCV, multiple R-squared, AIC, and BIC of fit model (assuming Gaussian error).
`myknots`	Spline knots used for fit.
`nvec`	Number of eigenvectors used for solution.
`rparm`	Rounding parameter for `x` (same as input).
`lambda`	Optimal smoothing parameter.
`coef`	Spline basis function coefficients.
`coef.csqrt`	Matrix square-root of covariace matrix of `coef`. Use `tcrossprod(coef.csqrt)` to get covariance matrix of `coef`.

Warnings

Input `nvec` must be greater than `ncol(x)+1`.

When using rounding parameters, output `fitted.values` corresponds to unique rounded predictor scores in output `xunique`. Use `predict.bigtps` function to get fitted values for full `y` vector.

Computational Details

According to thin-plate spline theory, the function η can be approximated as

$$\eta(x) = \sum_{k=1}^{M} d_k \phi_k(\mathbf{x}) + \sum_{h=1}^{q} c_h \xi(\mathbf{x}, \mathbf{x}_h^*)$$

where the $\{\phi_k\}_{k=1}^M$ are linear functions, ξ is the thin-plate spline semi-kernel, $\{\mathbf{x}_h^*\}_{h=1}^q$ are the knots, and the c_h coefficients are constrained to be orthongonal to the $\{\phi_k\}_{k=1}^M$ functions.

This implies that the penalized least-squares functional can be rewritten as

$$\|\mathbf{y} - \mathbf{Kd} - \mathbf{Jc}\|^2 + n\lambda \mathbf{c}'\mathbf{Qc}$$

where $\mathbf{K} = \{\phi(\mathbf{x}_i)\}_{n\times M}$ is the null space basis function matrix, $\mathbf{J} = \{\xi(\mathbf{x}_i, \mathbf{x}_h^*)\}_{n\times q}$ is the contrast space basis funciton matrix, $\mathbf{Q} = \{\xi(\mathbf{x}_g^*, \mathbf{x}_h^*)\}_{q\times q}$ is the penalty matrix, and $\mathbf{d} = (d_0, \ldots, d_M)'$ and $\mathbf{c} = (c_1, \ldots, c_q)'$ are the unknown basis function coefficients, where $\mathbf{c}$ are constrained to be orthongonal to the $\{\phi_k\}_{k=1}^M$ functions.

See Helwig and Ma for specifics about how the constrained estimation is handled.

Note

The spline is estimated using penalized least-squares, which does not require the Gaussian error assumption. However, the spline inference information (e.g., standard errors and fit information) requires the Gaussian error assumption.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define relatively smooth function
set.seed(773)
myfun <- function(x){ sin(2*pi*x) }
x <- runif(500)
y <- myfun(x) + rnorm(500)

# fit thin-plate spline (default 1 dim: 30 knots)
tpsmod <- bigtps(x,y)
tpsmod


##########   EXAMPLE 2   ##########

# define more jagged function
set.seed(773)
myfun <- function(x){ 2*x+cos(2*pi*x) }
x <- runif(500)*4
y <- myfun(x) + rnorm(500)

# try different numbers of knots
r1mod <- bigtps(x,y,nknots=20,rparm=0.01)
crossprod( myfun(r1mod$xunique) - r1mod$fitted )/length(r1mod$fitted)
r2mod <- bigtps(x,y,nknots=35,rparm=0.01)
crossprod( myfun(r2mod$xunique) - r2mod$fitted )/length(r2mod$fitted)
r3mod <- bigtps(x,y,nknots=50,rparm=0.01)
crossprod( myfun(r3mod$xunique) - r3mod$fitted )/length(r3mod$fitted)
```

```
##########   EXAMPLE 3   ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
x <- cbind(runif(500),runif(500))
y <- myfun(x[,1],x[,2]) + rnorm(500)

# fit thin-plate spline with 50 knots (default 2 dim: 100 knots)
tpsmod <- bigtps(x,y,nknots=50)
tpsmod
crossprod( myfun(x[,1],x[,2]) - tpsmod$fitted.values )/500


##########   EXAMPLE 4   ##########

# function with three continuous predictors
set.seed(773)
myfun <- function(x1v,x2v,x3v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*x3v)
  }
x <- cbind(runif(500),runif(500),runif(500))
y <- myfun(x[,1],x[,2],x[,3]) + rnorm(500)

# fit thin-plate spline with 50 knots (default 3 dim: 200 knots)
tpsmod <- bigtps(x,y,nknots=50)
tpsmod
crossprod( myfun(x[,1],x[,2],x[,3]) - tpsmod$fitted.values )/500
```

`binsamp` *Bin-Samples Strategic Knot Indices*

Description

Breaks the predictor domain into a user-specified number of disjoint subregions, and randomly samples a user-specified number of observations from each (nonempty) subregion.

Usage

```
binsamp(x,xrng=NULL,nmbin=11,nsamp=1,alg=c("new","old"))
```

Arguments

`x`	Matrix of predictors $\mathbf{X} = \{x_{ij}\}_{n \times p}$ where n is the number of observations, and p is the number of predictors.

xrng	Optional matrix of predictor ranges: $\mathbf{R} = \{r_{kj}\}_{2\times p}$ where $r_{1j} = \min_i x_{ij}$ and $r_{2j} = \max_i x_{ij}$.
nmbin	Vector $\mathbf{b} = (b_1, \ldots, b_p)'$, where $b_j \geq 1$ is the number of marginal bins to use for the j-th predictor. If `length(nmbin)<ncol(x)`, then `nmbin[1]` is used for all columns. Default is `nmbin=11` marginal bins for each dimension.
nsamp	Scalar $s \geq 1$ giving the number of observations to sample from each bin. Default is sample `nsamp=1` observation from each bin.
alg	Bin-sampling algorithm. New algorithm forms equidistant grid, whereas old algorithm forms approximately equidistant grid. New algorithm is default for versions 1.0-1 and later.

Value

Returns an index vector indicating the rows of x that were bin-sampled.

Warnings

If x_{ij} is nominal with g levels, the function requires $b_j = g$ and $x_{ij} \in \{1, \ldots, g\}$ for $i \in \{1, \ldots, n\}$.

Note

The number of returned knots will depend on the distribution of the covariate scores. The maximum number of possible bin-sampled knots is $s \prod_{j=1}^{p} b_j$, but fewer knots will be returned if one (or more) of the bins is empty (i.e., if there is no data in one or more bins).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
##########   EXAMPLE 1   ##########

# create 2-dimensional predictor (both continuous)
set.seed(123)
xmat <- cbind(runif(10^6),runif(10^6))

# Default use:
#   10 marginal bins for each predictor
#   sample 1 observation from each subregion
xind <- binsamp(xmat)

# get the corresponding knots
bknots <- xmat[xind,]

# compare to randomly-sampled knots
rknots <- xmat[sample(1:(10^6),100),]
par(mfrow=c(1,2))
plot(bknots,main="bin-sampled")
plot(rknots,main="randomly sampled")
```

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

# create 2-dimensional predictor (continuous and nominal)
set.seed(123)
xmat <- cbind(runif(10^6),sample(1:3,10^6,replace=TRUE))

# use 10 marginal bins for x1 and 3 marginal bins for x2
# and sample one observation from each subregion
xind <- binsamp(xmat,nmbin=c(10,3))

# get the corresponding knots
bknots <- xmat[xind,]

# compare to randomly-sampled knots
rknots <- xmat[sample(1:(10^6),30),]
par(mfrow=c(1,2))
plot(bknots,main="bin-sampled")
plot(rknots,main="randomly sampled")



##########   EXAMPLE 3   ##########

# create 3-dimensional predictor (continuous, continuous, nominal)
set.seed(123)
xmat <- cbind(runif(10^6),runif(10^6),sample(1:2,10^6,replace=TRUE))

# use 10 marginal bins for x1 and x2, and 2 marginal bins for x3
# and sample one observation from each subregion
xind <- binsamp(xmat,nmbin=c(10,10,2))

# get the corresponding knots
bknots <- xmat[xind,]

# compare to randomly-sampled knots
rknots <- xmat[sample(1:(10^6),200),]
par(mfrow=c(2,2))
plot(bknots[1:100,1:2],main="bin-sampled, x3=1")
plot(bknots[101:200,1:2],main="bin-sampled, x3=2")
plot(rknots[rknots[,3]==1,1:2],main="randomly sampled, x3=1")
plot(rknots[rknots[,3]==2,1:2],main="randomly sampled, x3=2")
```

---

imagebar *Displays a Color Image with Colorbar*

Description

This is a modification to the R function image that adds a colorbar to the margin.

Usage

```
imagebar(x,y,z,xlim=NULL,ylim=NULL,zlim=NULL,
         zlab=NULL,zcex.axis=NULL,zcex.lab=NULL,
         zaxis.at=NULL,zaxis.labels=TRUE,
         col=NULL,ncolor=21,drawbar=TRUE,zline=2,
         pltimage=c(.2,.8,.2,.8),pltbar=c(.82,.85,.2,.8),...)
```

Arguments

x, y	Locations of grid lines at which the values in z are measured. These must be finite, non-missing and in (strictly) ascending order.
z	A matrix containing the values to be plotted (NAs are allowed).
xlim, ylim	Ranges for the plotted x and y values, defaulting to the ranges of x and y.
zlim	The minimum and maximum z values for which colors should be plotted, defaulting to the range of the finite values of z.
zlab	Label for the colorbar.
zcex.axis	The magnification to be used for the z-axis annotation (colorbar scale).
zcex.lab	The magnification to be used for the z-axis label (zlab).
zaxis.at	The points at which tick-marks are to be drawn for the colorbar. Points outside of the range of zlim will not be plotted.
zaxis.labels	This can either be a logical value specifying whether (numerical) annotations are to be made at the tickmarks, or a character or expression vector of labels to be placed at the tickpoints.
col	Color scheme to use. Default is from blueviolet (low) to red (high).
ncolor	The number of colors to use in the color scheme.
drawbar	Logical indicating if the colorbar should be drawn.
zline	Number of lines into the margin at which the axis line will be drawn (see axis).
pltimage	A vector of the form c(x1, x2, y1, y2) giving the coordinates of the image region as fractions of the current figure region (see par).
pltbar	A vector of the form c(x1, x2, y1, y2) giving the coordinates of the colorbar region as fractions of the current figure region (see par).
...	Additional arguments to be passed to image (e.g., xlab, ylab, main, cex, cex.axis, cex.lab, etc.)

Value

Produces an image plot with a colorbar.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
##########   EXAMPLE 1   ##########

myfun <- function(x){
  2*sin(sqrt(x[,1]^2+x[,2]^2+.1))/sqrt(x[,1]^2+x[,2]^2+.1)
}
x <- expand.grid(seq(-8,8,l=100),seq(-8,8,l=100))
imagebar(seq(-8,8,l=100),seq(-8,8,l=100),matrix(myfun(x),100,100),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))),zlim=c(-0.5,2),zaxis.at=seq(-0.5,2,by=0.5))


##########   EXAMPLE 2   ##########

myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + 2*sin(sqrt(x2v^2+.1))/sqrt(x2v^2+.1)
}
x <- expand.grid(x1v=seq(0,1,l=100),x2v=seq(-8,8,l=100))
imagebar(seq(0,1,l=100),seq(-8,8,l=100),matrix(myfun(x$x1v,x$x2v),100,100),
         col=c("red","orange","yellow","white"),xlab="x1v",ylab="x2v",
         zlab=expression(hat(italic(y))),zlim=c(-1.5,3),zaxis.at=seq(-1.5,3,by=0.5))


##########   EXAMPLE 3   ##########

myfun <- function(x1v,x2v){
  sin(3*pi*x1v) + sin(2*pi*x2v) + 3*cos(pi*(x1v-x2v))
}
x <- expand.grid(x1v=seq(-1,1,l=100),x2v=seq(-1,1,l=100))
imagebar(seq(-1,1,l=100),seq(-1,1,l=100),matrix(myfun(x$x1v,x$x2v),100,100),
         col=c("blue","green","light green","yellow"),xlab="x1v",ylab="x2v",
         zlab=expression(hat(italic(y))),zlim=c(-5,5),zaxis.at=c(-5,0,5),
         zaxis.labels=c("low","med","high"))
```

---

makessa *Makes Objects to Fit Smoothing Spline ANOVA Models*

Description

This function creates a list containing the necessary information to fit a smoothing spline anova model (see bigssa).

Usage

```
makessa(formula,data=NULL,type=NULL,nknots=NULL,rparm=NA,
        lambdas=NULL,skip.iter=TRUE,se.fit=FALSE,rseed=1234,
        gcvopts=NULL,knotcheck=TRUE,gammas=NULL,weights=NULL,
        random=NULL,remlalg=c("FS","NR","EM","none"),remliter=500,
       remltol=10^-4,remltau=NULL)
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted (see Details and Examples for more information).
data	Optional data frame, list, or environment containing the variables in formula.
type	List of smoothing spline types for predictors in formula (see Details). Options include type="cub" for cubic, type="acub" for another cubic, type="per" for cubic periodic, type="tps" for cubic thin-plate, and type="nom" for nominal.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of data to use as knots.
rparm	List of rounding parameters for each predictor. See Details.
lambdas	Vector of global smoothing parameters to try. Default uses lambdas=10^-c(9:0)
skip.iter	Logical indicating whether to skip the iterative smoothing parameter update. Using skip.iter=FALSE should provide a more optimal solution, but the fitting time may be substantially longer. See Computational Details.
se.fit	Logical indicating if the standard errors of the fitted values should be estimated.
rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.
gcvopts	Control parameters for optimization. List with 3 elements: (a) maxit: maximum number of algorithm iterations, (b) gcvtol: covergence tolerance for iterative GCV update, and (c) alpha: tuning parameter for GCV minimization. Default: gcvopts=list(maxit=5,gcvtol=10^-5,alpha=1)
knotcheck	If TRUE, only unique knots are used (for stability).
gammas	List of initial smoothing parameters for each predictor. See Details.
weights	Vector of positive weights for fitting (default is vector of ones).
random	Adds random effects to model (see Random Effects section).
remlalg	REML algorithm for estimating variance components (see Random Effects section). Input is ignored if is.null(random).
remliter	Maximum number of iterations for REML estimation of variance components. Input is ignored if random=NULL.
remltol	Convergence tolerance for REML estimation of variance components. Input is ignored if random=NULL.
remltau	Initial estimate of variance parameters for REML estimation of variance components. Input is ignored if random=NULL.

Details

See bigssa and below example for more details.

Value

An object of class "makessa", which can be input to bigssa.

Warning

When inputting a "makessa" class object into bigssa, the formula input to bigssa must be a nested version of the original formula input to makessa. In other words, you cannot add any new effects after a "makessa" object has been created, but you can drop (remove) effects from the model.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE  ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
x1v <- runif(500)
x2v <- runif(500)
y <- myfun(x1v,x2v) + rnorm(500)

# fit 2 possible models (create information 2 separate times)
system.time({
  intmod <- bigssa(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
  addmod <- bigssa(y~x1v+x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
})

# fit 2 possible models (create information 1 time)
system.time({
  makemod <- makessa(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
  int2mod <- bigssa(y~x1v*x2v,makemod)
  add2mod <- bigssa(y~x1v+x2v,makemod)
})
```

```
# check difference (no difference)
crossprod( intmod$fitted.values - int2mod$fitted.values )
crossprod( addmod$fitted.values - add2mod$fitted.values )
```

---

makessg *Makes Objects to Fit Generalized Smoothing Spline ANOVA Models*

Description

This function creates a list containing the necessary information to fit a generalized smoothing spline anova model (see bigssg).

Usage

```
makessg(formula,family,data,type=NULL,nknots=NULL,rparm=NA,
        lambdas=NULL,skip.iter=TRUE,se.lp=FALSE,rseed=1234,
        gcvopts=NULL,knotcheck=TRUE,gammas=NULL,weights=NULL,
        gcvtype=c("acv","gacv","gacv.old"))
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted (see Details and Examples for more information).
family	Distribution for response. One of five options: "binomial", "poisson", "Gamma", "inverse.gaussian", or "negbin". See bigssg.
data	Optional data frame, list, or environment containing the variables in formula.
type	List of smoothing spline types for predictors in formula (see Details). Options include type="cub" for cubic, type="acub" for another cubic, type="per" for cubic periodic, type="tps" for cubic thin-plate, and type="nom" for nominal.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of data to use as knots.
rparm	List of rounding parameters for each predictor. See Details.
lambdas	Vector of global smoothing parameters to try. Default uses lambdas=10^-c(9:0)
skip.iter	Logical indicating whether to skip the iterative smoothing parameter update. Using skip.iter=FALSE should provide a more optimal solution, but the fitting time may be substantially longer. See Computational Details.
se.lp	Logical indicating if the standard errors of the linear predictors (η) should be estimated.
rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.

gcvopts	Control parameters for optimization. List with 6 elements: (i) maxit: maximum number of outer iterations, (ii) gcvtol: covergence tolerance for iterative GACV update, (iii) alpha: tuning parameter for GACV minimization, (iv) inmaxit: maximum number of inner iterations for iteratively reweighted fitting, (v) intol: inner convergence tolerance for iteratively reweighted fitting, and (vi) insub: number of data points to subsample when checking inner convergence. gcvopts=list(maxit=5,gcvtol=10^-5,alpha=1,inmaxit=100,intol=10^-5,insub=10^4)
knotcheck	If TRUE, only unique knots are used (for stability).
gammas	List of initial smoothing parameters for each predictor. See Details.
weights	Vector of positive weights for fitting (default is vector of ones).
gcvtype	Cross-validation criterion for selecting smoothing parameters (see Details).

Details

See bigssg and below example for more details.

Value

An object of class "makessg", which can be input to bigssg.

Warning

When inputting a "makessg" class object into bigssg, the formula input to bigssg must be a nested version of the original formula input to makessg. In other words, you cannot add any new effects after a "makessg" object has been created, but you can drop (remove) effects from the model.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Gu, C. and Xiang, D. (2001). Cross-validating non-Gaussian data: Generalized approximate cross-validation revisited. *Journal of Computational and Graphical Statistics, 10*, 581-591.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE  ##########

# function with two continuous predictors
set.seed(1)
```

```
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
ndpts <- 1000
x1v <- runif(ndpts)
x2v <- runif(ndpts)

# binomial response (no weights)
set.seed(773)
lp <- myfun(x1v,x2v)
p <- 1/(1+exp(-lp))
y <- rbinom(n=ndpts,size=1,p=p)

# fit 2 possible models (create information 2 separate times)
system.time({
  intmod <- bigssg(y~x1v*x2v,family="binomial",type=list(x1v="cub",x2v="cub"),nknots=50)
  addmod <- bigssg(y~x1v+x2v,family="binomial",type=list(x1v="cub",x2v="cub"),nknots=50)
})

# fit 2 possible models (create information 1 time)
system.time({
  makemod <- makessg(y~x1v*x2v,family="binomial",type=list(x1v="cub",x2v="cub"),nknots=50)
  int2mod <- bigssg(y~x1v*x2v,data=makemod)
  add2mod <- bigssg(y~x1v+x2v,data=makemod)
})

# check difference (no difference)
crossprod( intmod$fitted.values - int2mod$fitted.values )
crossprod( addmod$fitted.values - add2mod$fitted.values )
```

---

makessp *Makes Objects to Fit Smoothing Splines with Parametric Effects*

Description

This function creates a list containing the necessary information to fit a smoothing spline with parametric effects (see bigssp).

Usage

```
makessp(formula,data=NULL,type=NULL,nknots=NULL,rparm=NA,
        lambdas=NULL,skip.iter=TRUE,se.fit=FALSE,rseed=1234,
        gcvopts=NULL,knotcheck=TRUE,thetas=NULL,weights=NULL,
        random=NULL,remlalg=c("FS","NR","EM","none"),remliter=500,
       remltol=10^-4,remltau=NULL)
```

Arguments

formula	An object of class "formula": a symbolic description of the model to be fitted (see Details and Examples for more information).
data	Optional data frame, list, or environment containing the variables in formula.
type	List of smoothing spline types for predictors in formula (see Details). Options include type="cub" for cubic, type="acub" for another cubic, type="per" for cubic periodic, type="tps" for cubic thin-plate, and type="nom" for nominal. Use type="prm" for parametric effect.
nknots	Two possible options: (a) scalar giving total number of random knots to sample, or (b) vector indexing which rows of data to use as knots.
rparm	List of rounding parameters for each predictor. See Details.
lambdas	Vector of global smoothing parameters to try. Default uses lambdas=10^-c(9:0)
skip.iter	Logical indicating whether to skip the iterative smoothing parameter update. Using skip.iter=FALSE should provide a more optimal solution, but the fitting time may be substantially longer. See Computational Details.
se.fit	Logical indicating if the standard errors of the fitted values should be estimated.
rseed	Random seed for knot sampling. Input is ignored if nknots is an input vector of knot indices. Set rseed=NULL to obtain a different knot sample each time, or set rseed to any positive integer to use a different seed than the default.
gcvopts	Control parameters for optimization. List with 3 elements: (a) maxit: maximum number of algorithm iterations, (b) gcvtol: covergence tolerance for iterative GCV update, and (c) alpha: tuning parameter for GCV minimization. Default: gcvopts=list(maxit=5,gcvtol=10^-5,alpha=1)
knotcheck	If TRUE, only unique knots are used (for stability).
thetas	List of initial smoothing parameters for each predictor subspace. See Details.
weights	Vector of positive weights for fitting (default is vector of ones).
random	Adds random effects to model (see Random Effects section).
remlalg	REML algorithm for estimating variance components (see Random Effects section). Input is ignored if is.null(random).
remliter	Maximum number of iterations for REML estimation of variance components. Input is ignored if random=NULL.
remltol	Convergence tolerance for REML estimation of variance components. Input is ignored if random=NULL.
remltau	Initial estimate of variance parameters for REML estimation of variance components. Input is ignored if random=NULL.

Details

See bigssp and below example for more details.

Value

An object of class "makessp", which can be input to bigssp.

Warning

When inputting a "makessp" class object into bigssp, the formula input to bigssp must be a nested version of the original formula input to makessp. In other words, you cannot add any new effects after a "makessp" object has been created, but you can drop (remove) effects from the model.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE  ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
x1v <- runif(500)
x2v <- runif(500)
y <- myfun(x1v,x2v) + rnorm(500)

# fit 2 possible models (create information 2 separate times)
system.time({
  intmod <- bigssp(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
  addmod <- bigssp(y~x1v+x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
})

# fit 2 possible models (create information 1 time)
system.time({
  makemod <- makessp(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
  int2mod <- bigssp(y~x1v*x2v,makemod)
  add2mod <- bigssp(y~x1v+x2v,makemod)
})
```

```
# check difference (no difference)
crossprod( intmod$fitted.values - int2mod$fitted.values )
crossprod( addmod$fitted.values - add2mod$fitted.values )
```

`ordspline` *Fits Ordinal Smoothing Spline*

Description

Given a real-valued response vector $\mathbf{y} = \{y_i\}_{n\times 1}$ and an ordinal predictor vector $\mathbf{x} = \{x_i\}_{n\times 1}$ with $x_i \in \{1, \ldots, K\}\ \forall i$, an ordinal smoothing spline model has the form

$$y_i = \eta(x_i) + e_i$$

where y_i is the i-th observation's respone, x_i is the i-th observation's predictor, η is an unknown function relating the response and predictor, and $e_i \sim \mathrm{N}(0, \sigma^2)$ is iid Gaussian error.

Usage

```
ordspline(x, y, knots, weights, lambda, monotone=FALSE)
```

Arguments

`x`	Predictor vector.
`y`	Response vector. Must be same length as `x`.
`knots`	Either a scalar giving the number of equidistant knots to use, or a vector of values to use as the spline knots. If left blank, the number of knots is `min(50, nu)` where `nu = length(unique(x))`.
`weights`	Weights vector (for weighted penalized least squares). Must be same length as `x` and contain non-negative values.
`lambda`	Smoothing parameter. If left blank, `lambda` is tuned via Generalized Cross-Validation.
`monotone`	If `TRUE`, the relationship between `x` and `y` is constrained to be monotonic increasing.

Details

To estimate η I minimize the penalized least-squares functional

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \eta(x_i))^2 + \lambda \sum_{x=2}^{K}[\eta(x) - \eta(x-1)]^2 dx$$

where $\lambda \geq 0$ is a smoothing parameter that controls the trade-off between fitting and smoothing the data.

Default use of the function estimates λ by minimizing the GCV score:

$$\text{GCV}(\lambda) = \frac{n\|(\mathbf{I}_n - \mathbf{S}_\lambda)\mathbf{y}\|^2}{[n - \text{tr}(\mathbf{S}_\lambda)]^2}$$

where $\mathbf{I}_n$ is the identity matrix and $\mathbf{S}_\lambda$ is the smoothing matrix.

Value

`fitted.values`	Vector of fitted values.
`se.fit`	Vector of standard errors of `fitted.values`.
`sigma`	Estimated error standard deviation, i.e., $\hat{\sigma}$.
`lambda`	Chosen smoothing parameter.
`info`	Model fit information: vector containing the GCV, R-squared, AIC, and BIC of fit model (assuming Gaussian error).
`coef`	Spline basis function coefficients.
`coef.csqrt`	Matrix square-root of covariace matrix of `coef`. Use `tcrossprod(coef.csqrt)` to get covariance matrix of `coef`.
`n`	Number of data points, i.e., `length(x)`.
`df`	Effective degrees of freedom (trace of smoothing matrix).
`xunique`	Unique elements of `x`.
`x`	Predictor vector (same as input).
`y`	Response vector (same as input).
`residuals`	Residual vector, i.e., `y - fitted.values`.
`knots`	Spline knots used for fit.
`monotone`	Logical (same as input).

Warnings

When inputting user-specified `knots`, all values in `knots` must match a corresponding value in `x`.

Note

The spline is estimated using penalized least-squares, which does not require the Gaussian error assumption. However, the spline inference information (e.g., standard errors and fit information) requires the Gaussian error assumption.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. and 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*, 715-732.

Examples

```
##########   EXAMPLE   ##########

# generate some data
n <- 100
nk <- 50
x <- seq(-3,3,length.out=n)
eta <- (sin(2*x/pi) + 0.25*x^3 + 0.05*x^5)/15
set.seed(1)
y <- eta + rnorm(n, sd=0.5)

# plot data and true eta
plot(x, y)
lines(x, eta, col="blue", lwd=2)

# fit ordinal smoothing spline
ossmod <- ordspline(x, y, knots=nk)
lines(ossmod$x, ossmod$fit, col="red", lwd=2)

# fit monotonic smoothing spline
mssmod <- ordspline(x, y, knots=nk, monotone=TRUE)
lines(mssmod$x, mssmod$fit, col="purple", lwd=2)
```

---

plotbar *Generic X-Y Plotting with Colorbar*

Description

This is a modification to the R function `plot` that adds a colorbar to the margin.

Usage

```
plotbar(x,y,z,xlim=NULL,ylim=NULL,zlim=NULL,
        zlab=NULL,zcex.axis=NULL,zcex.lab=NULL,
        zaxis.at = NULL, zaxis.labels = TRUE,
        col=NULL,ncolor=21,drawbar=TRUE,zline=2,
        pltimage=c(.2,.8,.2,.8),pltbar=c(.82,.85,.2,.8),...)
```

Arguments

x, y	The x and y coordinates of the points to plot.
z	Numeric vector the same length as x and y containing the values to be plotted in color.
xlim, ylim	Ranges for the plotted x and y values, defaulting to the ranges of x and y.
zlim	The minimum and maximum z values for which colors should be plotted, defaulting to the range of the finite values of z.
zlab	Label for the colorbar.
zcex.axis	The magnification to be used for the z-axis annotation (colorbar scale).
zcex.lab	The magnification to be used for the z-axis label (zlab).
zaxis.at	The points at which tick-marks are to be drawn for the colorbar. Points outside of the range of zlim will not be plotted.
zaxis.labels	This can either be a logical value specifying whether (numerical) annotations are to be made at the tickmarks, or a character or expression vector of labels to be placed at the tickpoints.
col	Color scheme to use. Default is from blueviolet (low) to red (high).
ncolor	The number of colors to use in the color scheme.
drawbar	Logical indicating if the colorbar should be drawn.
zline	Number of lines into the margin at which the axis line will be drawn (see axis).
pltimage	A vector of the form c(x1, x2, y1, y2) giving the coordinates of the image region as fractions of the current figure region (see par).
pltbar	A vector of the form c(x1, x2, y1, y2) giving the coordinates of the colorbar region as fractions of the current figure region (see par).
...	Additional arguments to be passed to plot (e.g., xlab, ylab, main, cex, cex.axis, cex.lab, etc.)

Value

Produces a plot with a colorbar.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
##########   EXAMPLE 1   ##########

myfun <- function(x){
  2*sin(sqrt(x[,1]^2+x[,2]^2+.1))/sqrt(x[,1]^2+x[,2]^2+.1)
}
x <- expand.grid(seq(-8,8,l=100),seq(-8,8,l=100))
plotbar(x[,1],x[,2],myfun(x),
        xlab=expression(italic(x)[1]),
        ylab=expression(italic(x)[2]),
```

```
        zlab=expression(hat(italic(y))))


##########   EXAMPLE 2   ##########

myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + 2*sin(sqrt(x2v^2+.1))/sqrt(x2v^2+.1)
}
x <- expand.grid(x1v=seq(0,1,l=100),x2v=seq(-8,8,l=100))
plotbar(x[,1],x[,2],myfun(x$x1v,x$x2v),
         col=c("red","orange","yellow","white"),
         xlab="x1v",ylab="x2v",zlab=expression(hat(italic(y))))


##########   EXAMPLE 3   ##########

myfun <- function(x1v,x2v){
  sin(3*pi*x1v) + sin(2*pi*x2v) + 3*cos(pi*(x1v-x2v))
}
x <- expand.grid(x1v=seq(-1,1,l=100),x2v=seq(-1,1,l=100))
plotbar(x[,1],x[,2],myfun(x$x1v,x$x2v),
         col=c("blue","green","light green","yellow"),
         xlab="x1v",ylab="x2v",zlab=expression(hat(italic(y))))
```

`plotci` *Generic X-Y Plotting with Confidence Intervals*

Description

This is a modification to the R function `plot` that adds confidence intervals to the plot.

Usage

```
plotci(x, y, se, level=0.95, cval=NULL, col="blue",
       col.ci="cyan", alpha=0.65, add=FALSE,
       type="l", link=function(y){y}, axes=TRUE,
       bars=FALSE, barlty=1, barlwd=2, bw=0.2, ...)
```

Arguments

`x, y`	The x and y coordinates of the points to plot.
`se`	Numeric vector the same length as `x` and `y` containing the standard errors of the `y` values.
`level`	Significance level for the confidence interval. Default forms 95% interval.
`cval`	Critical value for the confidence interval. Default uses `cval=qnorm(1-(1-level)/2)`.
`col`	Color for plotting the relationship between `x` and `y`.
`col.ci`	Color for plotting the confidence interval.

alpha	Transparency used for plotting confidence polygons. Only used when bars=FALSE.
add	Logical indicating whether lines should be added to current plot.
type	Type of plot to create (defaults to "l" for lines).
link	Link function to apply. See Details.
axes	Logical indicating if the axes should be drawn.
bars	Logical indicating if confidence bars should be plotted instead of polygons.
barlty, barlwd	Line type and width for confidence bars. Only used when bars=TRUE.
bw	Positive scalar giving the width of the confidence bars. Only used when bars=TRUE.
...	Additional arguments to be passed to plot (e.g., xlab, ylab, main, cex, cex.axis, cex.lab, etc.)

Details

The plotted confidence interval is c(link(y-cval*se), link(y+cval*se)) where link is the user-specified link function and cval is the user-sepcified critival value, which defaults to cval = qnorm(1-(1-level)/2).

Value

Produces a plot with a colorbar.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
##########   EXAMPLE   ##########

# define relatively smooth function
set.seed(773)
myfun <- function(x){ sin(2*pi*x) }
x <- runif(10^4)
y <- myfun(x) + rnorm(10^4)

# fit cubic smoothing spline
cubmod <- bigspline(x,y)
newdata <- data.frame(x=seq(0,1,length=20))
ypred <- predict(cubmod, newdata, se.fit=TRUE)

# plot predictions with CIs in two ways
plotci(newdata$x, ypred$fit, ypred$se.fit)
plotci(newdata$x, ypred$fit, ypred$se.fit, type="p", bars=TRUE, bw=0.02)
```

`predict.bigspline` *Predicts for "bigspline" Objects*

Description

Get fitted values and standard error estimates for cubic smoothing splines.

Usage

```
## S3 method for class 'bigspline'
predict(object,newdata=NULL,se.fit=FALSE,
        effect=c("all","0","lin","non"),
        design=FALSE,smoothMatrix=FALSE,...)
```

Arguments

`object`	Object of class "bigspline", which is output from `bigspline`.
`newdata`	Vector containing new data points for prediction. See Details and Example. Default of `newdata=NULL` uses original data in `object` input.
`se.fit`	Logical indicating whether the standard errors of the fitted values should be estimated. Default is `se.fit=FALSE`.
`effect`	Which effect to estimate: `effect="all"` gives full $\hat{y}$, `effect="0"` gives the intercept (constant) portion of $\hat{y}$, `effect="lin"` gives linear portion of $\hat{y}$, and `effect="non"` gives nonlinear portion of $\hat{y}$.
`design`	Logical indicating whether the design matrix should be returned.
`smoothMatrix`	Logical indicating whether the smoothing matrix should be returned.
`...`	Ignored.

Details

Uses the coefficient and smoothing parameter estimates from a fit cubic smoothing spline (estimated by `bigspline`) to predict for new data.

Value

If `se.fit=FALSE`, `design=FALSE`, and `smoothMatrix=FALSE`, returns vector of fitted values.

Otherwise returns list with elements:

`fit`	Vector of fitted values
`se.fit`	Vector of standard errors of fitted values (if `se.fit=TRUE`)
`X`	Design matrix used to create fitted values (if `design=TRUE`)
`ix`	Index vector such that `fit=X%*%object$coef[ix]` (if `design=TRUE`)
`S`	Smoothing matrix corresponding to fitted values (if `smoothMatrix=TRUE`)

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define univariate function and data
set.seed(773)
myfun <- function(x){ 2 + x + sin(2*pi*x) }
x <- runif(10^4)
y <- myfun(x) + rnorm(10^4)

# fit cubic spline model
cubmod <- bigspline(x,y)
crossprod( predict(cubmod) - myfun(x) )/10^4

# define new data for prediction
newdata <- data.frame(x=seq(0,1,length.out=100))

# get fitted values and standard errors for new data
yc <- predict(cubmod,newdata,se.fit=TRUE)

# plot results with 95% Bayesian confidence interval
plot(newdata$x,yc$fit,type="l")
lines(newdata$x,yc$fit+qnorm(.975)*yc$se.fit,lty=3)
lines(newdata$x,yc$fit-qnorm(.975)*yc$se.fit,lty=3)

# predict constant, linear, and nonlinear effects
yc0 <- predict(cubmod,newdata,se.fit=TRUE,effect="0")
ycl <- predict(cubmod,newdata,se.fit=TRUE,effect="lin")
ycn <- predict(cubmod,newdata,se.fit=TRUE,effect="non")
crossprod( yc$fit - (yc0$fit + ycl$fit + ycn$fit) )

# plot results with 95% Bayesian confidence intervals
par(mfrow=c(1,2))
plot(newdata$x,ycl$fit,type="l",main="Linear effect")
```

```
lines(newdata$x,ycl$fit+qnorm(.975)*ycl$se.fit,lty=3)
lines(newdata$x,ycl$fit-qnorm(.975)*ycl$se.fit,lty=3)
plot(newdata$x,ycn$fit,type="l",main="Nonlinear effect")
lines(newdata$x,ycn$fit+qnorm(.975)*ycn$se.fit,lty=3)
lines(newdata$x,ycn$fit-qnorm(.975)*ycn$se.fit,lty=3)


##########   EXAMPLE 2   ##########

# define (same) univariate function and data
set.seed(773)
myfun <- function(x){ 2 + x + sin(2*pi*x) }
x <- runif(10^4)
y <- myfun(x) + rnorm(10^4)

# fit a different cubic spline model
cubamod <- bigspline(x,y,type="cub0")
crossprod( predict(cubamod) - myfun(x) )/10^4

# define (same) new data for prediction
newdata <- data.frame(x=seq(0,1,length.out=100))

# get fitted values and standard errors for new data
ya <- predict(cubamod,newdata,se.fit=TRUE)

# plot results with 95% Bayesian confidence interval
plot(newdata$x,ya$fit,type="l")
lines(newdata$x,ya$fit+qnorm(.975)*ya$se.fit,lty=3)
lines(newdata$x,ya$fit-qnorm(.975)*ya$se.fit,lty=3)

# predict constant, linear, and nonlinear effects
ya0 <- predict(cubamod,newdata,se.fit=TRUE,effect="0")
yal <- predict(cubamod,newdata,se.fit=TRUE,effect="lin")
yan <- predict(cubamod,newdata,se.fit=TRUE,effect="non")
crossprod( ya$fit - (ya0$fit + yal$fit + yan$fit) )

# plot results with 95% Bayesian confidence intervals
par(mfrow=c(1,2))
plot(newdata$x,yal$fit,type="l",main="Linear effect")
lines(newdata$x,yal$fit+qnorm(.975)*yal$se.fit,lty=3)
lines(newdata$x,yal$fit-qnorm(.975)*yal$se.fit,lty=3)
plot(newdata$x,yan$fit,type="l",main="Nonlinear effect")
lines(newdata$x,yan$fit+qnorm(.975)*yan$se.fit,lty=3)
lines(newdata$x,yan$fit-qnorm(.975)*yan$se.fit,lty=3)
```

`predict.bigssa` *Predicts for "bigssa" Objects*

Description

Get fitted values and standard error estimates for smoothing spline anova models.

Usage

```
## S3 method for class 'bigssa'
predict(object,newdata=NULL,se.fit=FALSE,include=object$tnames,
        effect=c("all","0","lin","non"),includeint=FALSE,
        design=FALSE,smoothMatrix=FALSE,intercept=NULL,...)
```

Arguments

object	Object of class "bigssa", which is output from bigssa.
newdata	Data frame or list containing the new data points for prediction. Variable names must match those used in the formula input of bigssa. See Details and Example. Default of newdata=NULL uses original data in object input.
se.fit	Logical indicating whether the standard errors of the fitted values should be estimated. Default is se.fit=FALSE.
include	Which terms to include in the estimate. You can get fitted values for any combination of terms in the tnames element of an "bigssa" object.
effect	Which effect to estimate: effect="all" gives $\hat{y}$ for given terms in include, effect="lin" gives linear portion of $\hat{y}$ for given terms in include, and effect="non" gives nonlinear portion of $\hat{y}$ for given terms in include. Use effect="0" to return the intercept.
includeint	Logical indicating whether the intercept should be included in the prediction. If include=object$tnames and effect="all" (default), then this input is ignored and the intercept is automatically included in the prediction.
design	Logical indicating whether the design matrix should be returned.
smoothMatrix	Logical indicating whether the smoothing matrix should be returned.
intercept	Logical indicating whether the intercept should be included in the prediction. When used, this input overrides the includeint input.
...	Ignored.

Details

Uses the coefficient and smoothing parameter estimates from a fit smoothing spline anova (estimated by bigssa) to predict for new data.

Value

If se.fit=FALSE, design=FALSE, and smoothMatrix=FALSE, returns vector of fitted values.

Otherwise returns list with elements:

fit	Vector of fitted values
se.fit	Vector of standard errors of fitted values (if se.fit=TRUE)
X	Design matrix used to create fitted values (if design=TRUE)

ix	Index vector such that fit=X%*%object$modelspec$coef[ix] (if design=TRUE)
S	Smoothing matrix corresponding to fitted values (if smoothMatrix=TRUE)

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define univariate function and data
set.seed(773)
myfun <- function(x){ 2 + x + sin(2*pi*x) }
x <- runif(500)
y <- myfun(x) + rnorm(500)

# fit cubic spline model
cubmod <- bigssa(y~x,type="cub",nknots=30)
crossprod( predict(cubmod) - myfun(x) )/500

# define new data for prediction
newdata <- data.frame(x=seq(0,1,length.out=100))

# get fitted values and standard errors for new data
yc <- predict(cubmod,newdata,se.fit=TRUE)

# plot results with 95% Bayesian confidence interval
plot(newdata$x,yc$fit,type="l")
lines(newdata$x,yc$fit+qnorm(.975)*yc$se.fit,lty=3)
lines(newdata$x,yc$fit-qnorm(.975)*yc$se.fit,lty=3)

# predict constant, linear, and nonlinear effects
yc0 <- predict(cubmod,newdata,se.fit=TRUE,effect="0")
ycl <- predict(cubmod,newdata,se.fit=TRUE,effect="lin")
```

```
ycn <- predict(cubmod,newdata,se.fit=TRUE,effect="non")
crossprod( yc$fit - (yc0$fit + ycl$fit + ycn$fit) )

# plot results with 95% Bayesian confidence intervals
par(mfrow=c(1,2))
plot(newdata$x,ycl$fit,type="l",main="Linear effect")
lines(newdata$x,ycl$fit+qnorm(.975)*ycl$se.fit,lty=3)
lines(newdata$x,ycl$fit-qnorm(.975)*ycl$se.fit,lty=3)
plot(newdata$x,ycn$fit,type="l",main="Nonlinear effect")
lines(newdata$x,ycn$fit+qnorm(.975)*ycn$se.fit,lty=3)
lines(newdata$x,ycn$fit-qnorm(.975)*ycn$se.fit,lty=3)


##########   EXAMPLE 2   ##########

# define bivariate function and data
set.seed(773)
myfun<-function(x){
  2 + x[,1]/10 - x[,2]/5 + 2*sin(sqrt(x[,1]^2+x[,2]^2+.1))/sqrt(x[,1]^2+x[,2]^2+.1)
}
x1v <- runif(500)*16-8
x2v <- runif(500)*16-8
y <- myfun(cbind(x1v,x2v)) + rnorm(500)

# tensor product cubic splines with 50 knots
cubmod <- bigssa(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=50)
crossprod( predict(cubmod) - myfun(cbind(x1v,x2v)) )/500

# define new data for prediction
xnew <- as.matrix(expand.grid(seq(-8,8,l=50),seq(-8,8,l=50)))
newdata <- list(x1v=xnew[,1],x2v=xnew[,2])

# get fitted values for new data
yp <- predict(cubmod,newdata)

# plot results
imagebar(seq(-8,8,l=50),seq(-8,8,l=50),matrix(yp,50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))))

# predict linear and nonlinear effects for x1v
newdata <- list(x1v=seq(-8,8,length.out=100))
yl <- predict(cubmod,newdata,include="x1v",effect="lin",se.fit=TRUE)
yn <- predict(cubmod,newdata,include="x1v",effect="non",se.fit=TRUE)

# plot results with 95% Bayesian confidence intervals
par(mfrow=c(1,2))
plot(newdata$x1v,yl$fit,type="l",main="Linear effect")
lines(newdata$x1v,yl$fit+qnorm(.975)*yl$se.fit,lty=3)
lines(newdata$x1v,yl$fit-qnorm(.975)*yl$se.fit,lty=3)
plot(newdata$x1v,yn$fit,type="l",main="Nonlinear effect",ylim=c(-.3,.4))
lines(newdata$x1v,yn$fit+qnorm(.975)*yn$se.fit,lty=3)
lines(newdata$x1v,yn$fit-qnorm(.975)*yn$se.fit,lty=3)
```

predict.bigssg *Predicts for "bigssg" Objects*

Description

Get fitted values and standard error estimates for generalized smoothing spline anova models.

Usage

```
## S3 method for class 'bigssg'
predict(object,newdata=NULL,se.lp=FALSE,include=object$tnames,
        effect=c("all","0","lin","non"),includeint=FALSE,
        design=FALSE,smoothMatrix=FALSE,intercept=NULL,...)
```

Arguments

object	Object of class "bigssg", which is output from bigssg.
newdata	Data frame or list containing the new data points for prediction. Variable names must match those used in the formula input of bigssg. See Details and Example. Default of newdata=NULL uses original data in object input.
se.lp	Logical indicating if the standard errors of the linear predictors (η) should be estimated. Default is se.lp=FALSE.
include	Which terms to include in the estimate. You can get fitted values for any combination of terms in the tnames element of an "bigssg" object.
effect	Which effect to estimate: effect="all" gives $\hat{y}$ for given terms in include, effect="lin" gives linear portion of $\hat{y}$ for given terms in include, and effect="non" gives nonlinear portion of $\hat{y}$ for given terms in include. Use effect="0" to return the intercept.
includeint	Logical indicating whether the intercept should be included in the prediction. If include=object$tnames and effect="all" (default), then this input is ignored and the intercept is automatically included in the prediction.
design	Logical indicating whether the design matrix should be returned.
smoothMatrix	Logical indicating whether the smoothing matrix should be returned.
intercept	Logical indicating whether the intercept should be included in the prediction. When used, this input overrides the includeint input.
...	Ignored.

Details

Uses the coefficient and smoothing parameter estimates from a fit generalized smoothing spline anova (estimated by bigssg) to predict for new data.

Value

Returns list with elements:

`fitted.values`	Vector of fitted values (on data scale)
`linear.predictors`	Vector of fitted values (on link scale)
`se.lp`	Vector of standard errors of linear predictors (if `se.lp=TRUE`)
`X`	Design matrix used to create linear predictors (if `design=TRUE`)
`ix`	Index vector such that `linear.predictors=X%*%object$modelspec$coef[ix]` (if `design=TRUE`)
`S`	Smoothing matrix corresponding to fitted values (if `smoothMatrix=TRUE`)

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Gu, C. and Xiang, D. (2001). Cross-validating non-Gaussian data: Generalized approximate cross-validation revisited. *Journal of Computational and Graphical Statistics, 10*, 581-591.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define univariate function and data
set.seed(1)
myfun <- function(x){ sin(2*pi*x) }
ndpts <- 1000
x <- runif(ndpts)

# negative binomial response (unknown dispersion)
set.seed(773)
lp <- myfun(x)
mu <- exp(lp)
y <- rnbinom(n=ndpts,size=2,mu=mu)

# fit cubic spline model
cubmod <- bigssg(y~x,family="negbin",type="cub",nknots=20)
1/cubmod$dispersion   ## dispersion = 1/size
crossprod( lp - cubmod$linear.predictor )/length(lp)
```

```
# define new data for prediction
newdata <- data.frame(x=seq(0,1,length.out=100))

# get fitted values and standard errors for new data
yc <- predict(cubmod,newdata,se.lp=TRUE)

# plot results with 95% Bayesian confidence interval (link scale)
plot(newdata$x,yc$linear.predictor,type="l")
lines(newdata$x,yc$linear.predictor+qnorm(.975)*yc$se.lp,lty=3)
lines(newdata$x,yc$linear.predictor-qnorm(.975)*yc$se.lp,lty=3)

# plot results with 95% Bayesian confidence interval (data scale)
plot(newdata$x,yc$fitted,type="l")
lines(newdata$x,exp(yc$linear.predictor+qnorm(.975)*yc$se.lp),lty=3)
lines(newdata$x,exp(yc$linear.predictor-qnorm(.975)*yc$se.lp),lty=3)

# predict constant, linear, and nonlinear effects
yc0 <- predict(cubmod,newdata,se.lp=TRUE,effect="0")
ycl <- predict(cubmod,newdata,se.lp=TRUE,effect="lin")
ycn <- predict(cubmod,newdata,se.lp=TRUE,effect="non")
crossprod( yc$linear - (yc0$linear + ycl$linear + ycn$linear) )

# plot results with 95% Bayesian confidence intervals (link scale)
par(mfrow=c(1,2))
plot(newdata$x,ycl$linear,type="l",main="Linear effect")
lines(newdata$x,ycl$linear+qnorm(.975)*ycl$se.lp,lty=3)
lines(newdata$x,ycl$linear-qnorm(.975)*ycl$se.lp,lty=3)
plot(newdata$x,ycn$linear,type="l",main="Nonlinear effect")
lines(newdata$x,ycn$linear+qnorm(.975)*ycn$se.lp,lty=3)
lines(newdata$x,ycn$linear-qnorm(.975)*ycn$se.lp,lty=3)

# plot results with 95% Bayesian confidence intervals (data scale)
par(mfrow=c(1,2))
plot(newdata$x,ycl$fitted,type="l",main="Linear effect")
lines(newdata$x,exp(ycl$linear+qnorm(.975)*ycl$se.lp),lty=3)
lines(newdata$x,exp(ycl$linear-qnorm(.975)*ycl$se.lp),lty=3)
plot(newdata$x,ycn$fitted,type="l",main="Nonlinear effect")
lines(newdata$x,exp(ycn$linear+qnorm(.975)*ycn$se.lp),lty=3)
lines(newdata$x,exp(ycn$linear-qnorm(.975)*ycn$se.lp),lty=3)



##########   EXAMPLE 2   ##########

# define bivariate function and data
set.seed(1)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
ndpts <- 1000
x1v <- runif(ndpts)
x2v <- runif(ndpts)
```

```
# binomial response (with weights)
set.seed(773)
lp <- myfun(x1v,x2v)
p <- 1/(1+exp(-lp))
w <- sample(c(10,20,30,40,50),length(p),replace=TRUE)
y <- rbinom(n=ndpts,size=w,p=p)/w   ## y is proportion correct
cubmod <- bigssg(y~x1v*x2v,family="binomial",type=list(x1v="cub",x2v="cub"),nknots=100,weights=w)
crossprod( lp - cubmod$linear.predictor )/length(lp)

# define new data for prediction
xnew <- as.matrix(expand.grid(seq(0,1,length=50),seq(0,1,length=50)))
newdata <- list(x1v=xnew[,1],x2v=xnew[,2])

# get fitted values for new data
yp <- predict(cubmod,newdata)

# plot linear predictor and fitted values
par(mfrow=c(2,2))
imagebar(seq(0,1,l=50),seq(0,1,l=50),matrix(myfun(newdata$x1v,newdata$x2v),50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))),zlim=c(-4.5,1.5),main="True Linear Predictor")
imagebar(seq(0,1,l=50),seq(0,1,l=50),matrix(yp$linear.predictor,50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
       zlab=expression(hat(italic(y))),zlim=c(-4.5,1.5),main="Estimated Linear Predictor")
newprob <- 1/(1+exp(-myfun(newdata$x1v,newdata$x2v)))
imagebar(seq(0,1,l=50),seq(0,1,l=50),matrix(newprob,50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))),zlim=c(0,0.8),main="True Probabilities")
imagebar(seq(0,1,l=50),seq(0,1,l=50),matrix(yp$fitted.values,50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))),zlim=c(0,0.8),main="Estimated Probabilities")

# predict linear and nonlinear effects for x1v (link scale)
newdata <- list(x1v=seq(0,1,length.out=100))
yl <- predict(cubmod,newdata,include="x1v",effect="lin",se.lp=TRUE)
yn <- predict(cubmod,newdata,include="x1v",effect="non",se.lp=TRUE)

# plot results with 95% Bayesian confidence intervals (link scale)
par(mfrow=c(1,2))
plot(newdata$x1v,yl$linear,type="l",main="Linear effect")
lines(newdata$x1v,yl$linear+qnorm(.975)*yl$se.lp,lty=3)
lines(newdata$x1v,yl$linear-qnorm(.975)*yl$se.lp,lty=3)
plot(newdata$x1v,yn$linear,type="l",main="Nonlinear effect")
lines(newdata$x1v,yn$linear+qnorm(.975)*yn$se.lp,lty=3)
lines(newdata$x1v,yn$linear-qnorm(.975)*yn$se.lp,lty=3)
```

`predict.bigssp` *Predicts for "bigssp" Objects*

Description

Get fitted values and standard error estimates for smoothing splines with parametric effects.

Usage

```
## S3 method for class 'bigssp'
predict(object,newdata=NULL,se.fit=FALSE,include=object$tnames,
        effect=c("all","0","lin","non"),includeint=FALSE,
        design=FALSE,smoothMatrix=FALSE,intercept=NULL,...)
```

Arguments

`object`	Object of class "bigssp", which is output from `bigssp`.
`newdata`	Data frame or list containing the new data points for prediction. Variable names must match those used in the `formula` input of `bigssp`. See Details and Example. Default of `newdata=NULL` uses original data in `object` input.
`se.fit`	Logical indicating whether the standard errors of the fitted values should be estimated. Default is `se.fit=FALSE`.
`include`	Which terms to include in the estimate. You can get fitted values for any combination of terms in the `tnames` element of an "bigssp" object.
`effect`	Which effect to estimate: `effect="all"` gives $\hat{y}$ for given terms in `include`, `effect="lin"` gives linear portion of $\hat{y}$ for given terms in `include`, and `effect="non"` gives nonlinear portion of $\hat{y}$ for given terms in `include`. Use `effect="0"` to return the intercept.
`includeint`	Logical indicating whether the intercept should be included in the prediction. If `include=object$tnames` and `effect="all"` (default), then this input is ignored and the intercept is automatically included in the prediction.
`design`	Logical indicating whether the design matrix should be returned.
`smoothMatrix`	Logical indicating whether the smoothing matrix should be returned.
`intercept`	Logical indicating whether the intercept should be included in the prediction. When used, this input overrides the `includeint` input.
`...`	Ignored.

Details

Uses the coefficient and smoothing parameter estimates from a fit smoothing spline with parametric effects (estimated by `bigssp`) to predict for new data.

Value

If `se.fit=FALSE`, `design=FALSE`, and `smoothMatrix=FALSE`, returns vector of fitted values.

Otherwise returns list with elements:

`fit`	Vector of fitted values
`se.fit`	Vector of standard errors of fitted values (if `se.fit=TRUE`)
`X`	Design matrix used to create fitted values (if `design=TRUE`)

ix	Index vector such that fit=X%*%object$modelspec$coef[ix] (if design=TRUE)
S	Smoothing matrix corresponding to fitted values (if smoothMatrix=TRUE)

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

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*, 383-398.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. (2016). Efficient estimation of variance components in nonparametric mixed-effects models with large samples. *Statistics and Computing, 26*, 1319-1336.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define univariate function and data
set.seed(773)
myfun <- function(x){ 2 + x + sin(2*pi*x) }
x <- runif(500)
y <- myfun(x) + rnorm(500)

# fit cubic spline model
cubmod <- bigssp(y~x,type="cub",nknots=30)
crossprod( predict(cubmod) - myfun(x) )/500

# define new data for prediction
newdata <- data.frame(x=seq(0,1,length.out=100))

# get fitted values and standard errors for new data
yc <- predict(cubmod,newdata,se.fit=TRUE)

# plot results with 95% Bayesian confidence interval
plot(newdata$x,yc$fit,type="l")
lines(newdata$x,yc$fit+qnorm(.975)*yc$se.fit,lty=3)
lines(newdata$x,yc$fit-qnorm(.975)*yc$se.fit,lty=3)
```

```
# predict constant, linear, and nonlinear effects
yc0 <- predict(cubmod,newdata,se.fit=TRUE,effect="0")
ycl <- predict(cubmod,newdata,se.fit=TRUE,effect="lin")
ycn <- predict(cubmod,newdata,se.fit=TRUE,effect="non")
sum( yc$fit - (yc0$fit + ycl$fit + ycn$fit) )

# plot results with 95% Bayesian confidence intervals
par(mfrow=c(1,2))
plot(newdata$x,ycl$fit,type="l",main="Linear effect")
lines(newdata$x,ycl$fit+qnorm(.975)*ycl$se.fit,lty=3)
lines(newdata$x,ycl$fit-qnorm(.975)*ycl$se.fit,lty=3)
plot(newdata$x,ycn$fit,type="l",main="Nonlinear effect")
lines(newdata$x,ycn$fit+qnorm(.975)*ycn$se.fit,lty=3)
lines(newdata$x,ycn$fit-qnorm(.975)*ycn$se.fit,lty=3)


##########   EXAMPLE 2   ##########

# define bivariate function and data
set.seed(773)
myfun <- function(x){
  2 + x[,1]/10 - x[,2]/5 + 2*sin(sqrt(x[,1]^2+x[,2]^2+.1))/sqrt(x[,1]^2+x[,2]^2+.1)
}
x <- cbind(runif(500),runif(500))*16 - 8
y <- myfun(x)+rnorm(500)

# bidimensional thin-plate spline with 50 knots
tpsmod <- bigssp(y~x,type="tps",nknots=50)
crossprod( predict(tpsmod) - myfun(x) )/500

# define new data for prediction
xnew <- as.matrix(expand.grid(seq(-8,8,length=50),seq(-8,8,length=50)))
newdata <- list(x=xnew)

# get fitted values for new data
yp <- predict(tpsmod,newdata)

# plot results
imagebar(seq(-8,8,l=50),seq(-8,8,l=50),matrix(yp,50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))))

# predict linear and nonlinear effects
yl <- predict(tpsmod,newdata,effect="lin")
yn <- predict(tpsmod,newdata,effect="non")

# plot results
par(mfrow=c(1,2))
imagebar(seq(-8,8,l=50),seq(-8,8,l=50),matrix(yl,50,50),
         main="Linear effect",xlab=expression(italic(x)[1]),
         ylab=expression(italic(x)[2]),zlab=expression(hat(italic(y))))
imagebar(seq(-8,8,l=50),seq(-8,8,l=50),matrix(yn,50,50),
         main="Nonlinear effect",xlab=expression(italic(x)[1]),
```

```
ylab=expression(italic(x)[2]),zlab=expression(hat(italic(y))))
```

`predict.bigtps` *Predicts for "bigtps" Objects*

Description

Get fitted values and standard error estimates for thin-plate splines.

Usage

```
## S3 method for class 'bigtps'
predict(object,newdata=NULL,se.fit=FALSE,
        effect=c("all","0","lin","non"),
        design=FALSE,smoothMatrix=FALSE,...)
```

Arguments

`object`	Object of class "bigtps", which is output from `bigtps`.
`newdata`	Vector or matrix containing new data points for prediction. See Details and Example. Default of `newdata=NULL` uses original data in `object` input.
`se.fit`	Logical indicating whether the standard errors of the fitted values should be estimated. Default is `se.fit=FALSE`.
`effect`	Which effect to estimate: `effect="all"` gives full $\hat{y}$, `effect="0"` gives the intercept (constant) portion of $\hat{y}$, `effect="lin"` gives linear portion of $\hat{y}$, and `effect="non"` gives nonlinear portion of $\hat{y}$.
`design`	Logical indicating whether the design matrix should be returned.
`smoothMatrix`	Logical indicating whether the smoothing matrix should be returned.
`...`	Ignored.

Details

Uses the coefficient and smoothing parameter estimates from a fit thin-plate spline (estimated by `bigtps`) to predict for new data.

Value

If `se.fit=FALSE`, `design=FALSE`, and `smoothMatrix=FALSE`, returns vector of fitted values.

Otherwise returns list with elements:

`fit`	Vector of fitted values
`se.fit`	Vector of standard errors of fitted values (if `se.fit=TRUE`)
`X`	Design matrix used to create fitted values (if `design=TRUE`)
`ix`	Index vector such that `fit=X%*%object$coef[ix]` (if `design=TRUE`)
`S`	Smoothing matrix corresponding to fitted values (if `smoothMatrix=TRUE`)

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. and 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*, 715-732.

Helwig, N. E. and Ma, P. (2016). Smoothing spline ANOVA for super-large samples: Scalable computation via rounding parameters. *Statistics and Its Interface, 9*, 433-444.

Examples

```
##########   EXAMPLE 1   ##########

# define univariate function and data
set.seed(773)
myfun <- function(x){ 2 + x + sin(2*pi*x) }
x <- runif(10^4)
y <- myfun(x) + rnorm(10^4)

# fit thin-plate spline (default 1 dim: 30 knots)
tpsmod <- bigtps(x,y)
crossprod( predict(tpsmod) - myfun(x) )/10^4

# define new data for prediction
newdata <- data.frame(x=seq(0,1,length.out=100))

# get fitted values and standard errors for new data
yc <- predict(tpsmod,newdata,se.fit=TRUE)

# plot results with 95% Bayesian confidence interval
plot(newdata$x,yc$fit,type="l")
lines(newdata$x,yc$fit+qnorm(.975)*yc$se.fit,lty=3)
lines(newdata$x,yc$fit-qnorm(.975)*yc$se.fit,lty=3)

# predict constant, linear, and nonlinear effects
yc0 <- predict(tpsmod,newdata,se.fit=TRUE,effect="0")
ycl <- predict(tpsmod,newdata,se.fit=TRUE,effect="lin")
ycn <- predict(tpsmod,newdata,se.fit=TRUE,effect="non")
crossprod( yc$fit - (yc0$fit + ycl$fit + ycn$fit) )

# plot results with 95% Bayesian confidence intervals
par(mfrow=c(1,2))
plot(newdata$x,ycl$fit,type="l",main="Linear effect")
lines(newdata$x,ycl$fit+qnorm(.975)*ycl$se.fit,lty=3)
lines(newdata$x,ycl$fit-qnorm(.975)*ycl$se.fit,lty=3)
plot(newdata$x,ycn$fit,type="l",main="Nonlinear effect")
lines(newdata$x,ycn$fit+qnorm(.975)*ycn$se.fit,lty=3)
```

```
lines(newdata$x,ycn$fit-qnorm(.975)*ycn$se.fit,lty=3)


##########   EXAMPLE 2   ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
x <- cbind(runif(10^4),runif(10^4))
y <- myfun(x[,1],x[,2]) + rnorm(10^4)

# fit thin-plate spline (default 2 dim: 100 knots)
tpsmod <- bigtps(x,y)

# define new data
newdata <- as.matrix(expand.grid(seq(0,1,length=50),seq(0,1,length=50)))

# get fitted values for new data
yp <- predict(tpsmod,newdata)

# plot results
imagebar(seq(0,1,length=50),seq(0,1,length=50),matrix(yp,50,50),
         xlab=expression(italic(x)[1]),ylab=expression(italic(x)[2]),
         zlab=expression(hat(italic(y))))

# predict linear and nonlinear effects
yl <- predict(tpsmod,newdata,effect="lin")
yn <- predict(tpsmod,newdata,effect="non")

# plot results
par(mfrow=c(1,2))
imagebar(seq(0,1,length=50),seq(0,1,length=50),matrix(yl,50,50),
         main="Linear effect",xlab=expression(italic(x)[1]),
         ylab=expression(italic(x)[2]),zlab=expression(hat(italic(y))))
imagebar(seq(0,1,length=50),seq(0,1,length=50),matrix(yn,50,50),
         main="Nonlinear effect",xlab=expression(italic(x)[1]),
         ylab=expression(italic(x)[2]),zlab=expression(hat(italic(y))))
```

`predict.ordspline` *Predicts for "ordspline" Objects*

Description

Get fitted values and standard error estimates for ordinal smoothing splines.

Usage

```
## S3 method for class 'ordspline'
predict(object,newdata=NULL,se.fit=FALSE,...)
```

Arguments

object	Object of class "ordspline", which is output from ordspline.
newdata	Vector containing new data points for prediction. See Details and Example. Default of newdata=NULL uses original data in object input.
se.fit	Logical indicating whether the standard errors of the fitted values should be estimated. Default is se.fit=FALSE.
...	Ignored.

Details

Uses the coefficient and smoothing parameter estimates from a fit ordinal smoothing spline (estimated by ordspline) to predict for new data.

Value

If se.fit=FALSE, returns vector of fitted values.

Otherwise returns list with elements:

fit	Vector of fitted values
se.fit	Vector of standard errors of fitted values

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Gu, C. (2013). *Smoothing spline ANOVA models, 2nd edition*. New York: Springer.

Helwig, N. E. (2013). *Fast and stable smoothing spline analysis of variance models for large samples with applications to electroencephalography data analysis*. Unpublished doctoral dissertation. University of Illinois at Urbana-Champaign.

Helwig, N. E. and 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*, 715-732.

Examples

```
##########   EXAMPLE   ##########

# define univariate function and data
set.seed(773)
myfun <- function(x){ 2 + x/2 + sin(x) }
```

```
x <- sample(1:20, size=500, replace=TRUE)
y <- myfun(x) + rnorm(500)

# fit ordinal spline model
ordmod <- ordspline(x, y)
monmod <- ordspline(x, y, monotone=TRUE)
crossprod( predict(ordmod) - myfun(x) ) / 500
crossprod( predict(monmod) - myfun(x) ) / 500

# plot truth and predictions
ordfit <- predict(ordmod, 1:20, se.fit=TRUE)
monfit <- predict(monmod, 1:20, se.fit=TRUE)
plotci(1:20, ordfit$fit, ordfit$se.fit, ylab="f(x)")
plotci(1:20, monfit$fit, monfit$se.fit, col="red", col.ci="pink", add=TRUE)
points(1:20, myfun(1:20))
```

---

print *Prints Fit Information for bigsplines Model*

Description

This function prints basic model fit information for a fit `bigsplines` model.

Usage

```
## S3 method for class 'bigspline'
print(x,...)
## S3 method for class 'bigssa'
print(x,...)
## S3 method for class 'bigssg'
print(x,...)
## S3 method for class 'bigssp'
print(x,...)
## S3 method for class 'bigtps'
print(x,...)
## S3 method for class 'ordspline'
print(x,...)
## S3 method for class 'summary.bigspline'
print(x,digits=4,...)
## S3 method for class 'summary.bigssa'
print(x,digits=4,...)
## S3 method for class 'summary.bigssg'
print(x,digits=4,...)
## S3 method for class 'summary.bigssp'
print(x,digits=4,...)
## S3 method for class 'summary.bigtps'
print(x,digits=4,...)
```

Arguments

x	Object of class "bigspline" (output from bigspline), class "summary.bigspline" (output from summary.bigspline), class "bigssa" (output from bigssa), class "summary.bigssa" (output from summary.bigssa), class "bigssg" (output from bigssg), class "summary.bigssg" (output from summary.bigssg), class "bigssp" (output from bigssp), class "summary.bigssp" (output from summary.bigssp), class "bigtps" (output from bigtps), class "summary.bigtps" (output from summary.bigtps), or class "ordspline" (output from ordspline).
digits	Number of decimal places to print.
...	Ignored.

Details

See bigspline, bigssa, bigssg, bigssp, bigtps, and ordspline for more details.

Value

"bigspline" objects: prints Spline Type, Fit Statistic information, and Smoothing Parameter.

"summary.bigspline" objects: prints Spline Type, five number summary of Residuals, Error Standard Deviation Estimate, Fit Statistics, and Smoothing Parameter.

"bigssa" objects: prints Spline Types, Fit Statistic information, and Algorithm Convergence status.

"summary.bigssa" objects: prints the formula Call, five number summary of Residuals, Error Standard Deviation Estimate, Fit Statistics, and Smoothing Parameters.

"bigssg" objects: prints Family, Spline Types, Fit Statistic information, and Algorithm Convergence status.

"summary.bigssg" objects: prints the Family, formula Call, five number summary of Residuals, Dispersion Estimate, Fit Statistics, and Smoothing Parameters (with selection criterion).

"bigssp" objects: prints Predictor Types, Fit Statistic information, and Algorithm Convergence status.

"summary.bigssp" objects: prints formula Call, five number summary of Residuals, Error Standard Deviation Estimate, Fit Statistics, and Smoothing Parameters.

"bigtps" objects: prints Spline Type, Fit Statistic information, and Smoothing Parameter.

"summary.bigtps" objects: prints Spline Type, five number summary of Residuals, Error Standard Deviation Estimate, Fit Statistics, and Smoothing Parameter.

"ordspline" objects: prints Monotonic, Fit Statistic information, and Smoothing Parameter.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
### see examples for bigspline, bigssa, bigssg, bigssp, bigtps, and ordspline
```

ssBasis *Smoothing Spline Basis for Polynomial Splines*

Description

Generate the smoothing spline basis matrix for a polynomial spline.

Usage

```
ssBasis(x, knots, m=2, d=0, xmin=min(x), xmax=max(x), periodic=FALSE, intercept=FALSE)
```

Arguments

x	Predictor variable.
knots	Spline knots.
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].
intercept	If TRUE, the first column of the basis will be a column of ones.

Value

X	Spline Basis.
knots	Spline knots.
m	Penalty order.
d	Derivative order.
xlim	Inputs xmin and xmax.
periodic	Same as input.
intercept	Same as input.

Note

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

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
##########   EXAMPLE   ##########

# define function and its derivatives
n <- 500
x <- seq(0, 1, length.out=n)
knots <- seq(0, 1, length=20)
y <- sin(4 * pi * x)
d1y <- 4 * pi * cos(4 * pi * x)
d2y <- - (4 * pi)^2 * sin(4 * pi * x)

# linear smoothing spline
linmat0 <- ssBasis(x, knots, m=1)
lincoef <- pinvsm(crossprod(linmat0$X)) %*% crossprod(linmat0$X, y)
linyhat <- linmat0$X %*% lincoef
linmat1 <- ssBasis(x, knots, m=1, d=1)
linyd1 <- linmat1$X %*% lincoef

# plot linear smoothing spline results
par(mfrow=c(1,2))
plot(x, y, type="l", main="Function")
lines(x, linyhat, lty=2, col="red")
plot(x, d1y, type="l", main="First Derivative")
lines(x, linyd1, lty=2, col="red")

# cubic smoothing spline
cubmat0 <- ssBasis(x, knots)
cubcoef <- pinvsm(crossprod(cubmat0$X)) %*% crossprod(cubmat0$X, y)
cubyhat <- cubmat0$X %*% cubcoef
cubmat1 <- ssBasis(x, knots, d=1)
cubyd1 <- cubmat1$X %*% cubcoef
cubmat2 <- ssBasis(x, knots, d=2)
cubyd2 <- cubmat2$X %*% cubcoef

# plot cubic smoothing spline results
par(mfrow=c(1,3))
plot(x, y, type="l", main="Function")
lines(x, cubyhat, lty=2, col="red")
plot(x, d1y, type="l", main="First Derivative")
lines(x, cubyd1, lty=2, col="red")
plot(x, d2y, type="l", main="Second Derivative")
lines(x, cubyd2, lty=2, col="red")

# quintic smoothing spline
quimat0 <- ssBasis(x, knots, m=3)
quicoef <- pinvsm(crossprod(quimat0$X)) %*% crossprod(quimat0$X, y)
quiyhat <- quimat0$X %*% quicoef
quimat1 <- ssBasis(x, knots, m=3, d=1)
quiyd1 <- quimat1$X %*% quicoef
quimat2 <- ssBasis(x, knots, m=3, d=2)
quiyd2 <- quimat2$X %*% quicoef
```

```
# plot quintic smoothing spline results
par(mfrow=c(1,3))
plot(x, y, type="l", main="Function")
lines(x, quiyhat, lty=2, col="red")
plot(x, d1y, type="l", main="First Derivative")
lines(x, quiyd1, lty=2, col="red")
plot(x, d2y, type="l", main="Second Derivative")
lines(x, quiyd2, lty=2, col="red")
```

---

summary *Summarizes Fit Information for bigsplines Model*

Description

This function summarizes basic model fit information for a fit `bigsplines` model.

Usage

```
## S3 method for class 'bigspline'
summary(object,fitresid=TRUE,chunksize=10000,...)
## S3 method for class 'bigssa'
summary(object,fitresid=TRUE,chunksize=10000,...)
## S3 method for class 'bigssg'
summary(object,fitresid=TRUE,chunksize=10000,...)
## S3 method for class 'bigssp'
summary(object,fitresid=TRUE,chunksize=10000,...)
## S3 method for class 'bigtps'
summary(object,fitresid=TRUE,chunksize=10000,...)
```

Arguments

`object`	Object of class "bigspline" (output from `bigspline`), class "bigssa" (output from `bigssa`), class "bigssg" (output from `bigssg`), class "bigssp" (output from `bigssp`), or class "bigtps" (output from `bigtps`).
`fitresid`	Logical indicating whether the fitted values and residuals should be calculated for all data points in input `object`.
`chunksize`	If `fitresid=TRUE`, fitted values are calculated in chunks of size `chunksize`.
`...`	Ignored.

Details

See `bigspline`, `bigssa`, `bigssg`, `bigssp`, and `bigtps` for more details.

Value

call	Called model in input formula.
type	Type of smoothing spline that was used for each predictor.
fitted.values	Vector of fitted values (if fitresid=TRUE).
linear.predictors	Vector of linear predictors (only for class "bigssg" with fitresid=TRUE).
residuals	Vector of residuals (if fitresid=TRUE). For class "bigssg" these are deviance residuals.
sigma	Estimated error standard deviation.
deviance	Model deviance (only for class "bigssg").
dispersion	Estimated dispersion parameter (only for class "bigssg").
n	Total sample size.
df	Effective degrees of freedom of the model.
info	Model fit information: vector containing the GCV, multiple R-squared, AIC, and BIC of fit model.
converged	Convergence status: converged=TRUE if the iterative theta update converged, converged=FALSE if the iterative theta update failed to converge, and converged=NA if option skip.iter=TRUE was used.
iter	Number of iterative updates (iter=NA if option skip.iter=TRUE was used).
rparm	Rounding parameters used for model fitting.
lambda	Global smoothing parameter used for model fitting.
gammas	Vector of additional smoothing parameters (only for class "bigssa").
thetas	Vector of additional smoothing parameters (only for class "bigssp").
family	Distribution family (only for class "bigssg").
gcvtype	Smoothing parameter selection criterion (only for class "bigssg").

Note

For "bigspline" and "bigtps" objects, the outputs call, converged, and iter are NA.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Examples

```
##########   EXAMPLE 1   ##########

# define relatively smooth function
set.seed(773)
myfun <- function(x){ sin(2*pi*x) }
x <- runif(10^4)
y <- myfun(x) + rnorm(10^4)
```

```
# cubic spline
cubmod <- bigspline(x,y)
summary(cubmod)


##########   EXAMPLE 2   ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
x1v <- runif(10^4)
x2v <- runif(10^4)
y <- myfun(x1v,x2v) + rnorm(10^4)

# cubic splines with 100 randomly selected knots (efficient parameterization)
cubmod <- bigssa(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=100)
summary(cubmod)


##########   EXAMPLE 3   ##########

# function with two continuous predictors
set.seed(1)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
ndpts <- 1000
x1v <- runif(ndpts)
x2v <- runif(ndpts)

# poisson response
set.seed(773)
lp <- myfun(x1v,x2v)
mu <- exp(lp)
y <- rpois(n=ndpts,lambda=mu)

# generalized smoothing spline anova
genmod <- bigssg(y~x1v*x2v,family="poisson",type=list(x1v="cub",x2v="cub"),nknots=50)
summary(genmod)


##########   EXAMPLE 4   ##########

# function with two continuous predictors
set.seed(773)
myfun <- function(x1v,x2v){
  sin(2*pi*x1v) + log(x2v+.1) + cos(pi*(x1v-x2v))
}
x1v <- runif(10^4)
x2v <- runif(10^4)
```

```
y <- myfun(x1v,x2v) + rnorm(10^4)

# cubic splines with 100 randomly selected knots (classic parameterization)
cubmod <- bigssp(y~x1v*x2v,type=list(x1v="cub",x2v="cub"),nknots=100)
summary(cubmod)


##########   EXAMPLE 5   ##########

# define relatively smooth function
set.seed(773)
myfun <- function(x){ sin(2*pi*x) }
x <- runif(10^4)
y <- myfun(x) + rnorm(10^4)

# thin-plate with default (30 knots)
tpsmod <- bigtps(x,y)
summary(tpsmod)
```

Index

∗Topic **package**
bigsplines-package, 2

axis, *33*, *45*

bigspline, *2*, 3, *9, 10*, *15*, *23, 24*, *27*, *48*, *66*, *69*
bigsplines *(*bigsplines-package*)*, 2
bigsplines-package, 2
bigssa, *2*, 8, *34–36*, *51*, *66*, *69*
bigssg, *2*, 14, *37, 38*, *54*, *66*, *69*
bigssp, *2*, 21, *39–41*, *58*, *66*, *69*
bigtps, *2*, 26, *61*, *66*, *69*
binsamp, 30

image, *33*
imagebar, 32

lm, *9*, *15*, *22*

makessa, *8*, 34
makessg, *14*, 37
makessp, *22*, 39

nlm, *11*, *18*, *24*

ordspline, 42, *64*, *66*

par, *33*, *45*
plot, *44*, *46*
plotbar, 44
plotci, 46
predict.bigspline, *5, 6*, 48
predict.bigssa, *10*, 50
predict.bigssg, *16*, 54
predict.bigssp, *23*, 57
predict.bigtps, *28*, 61
predict.ordspline, 63
print, 65

set.seed, *4*

ssBasis, 67
summary, 69
summary.bigspline, *66*
summary.bigssa, *66*
summary.bigssg, *66*
summary.bigssp, *66*
summary.bigtps, *66*