Package 'SOP'

January 20, 2025

Type Package

Title Generalised Additive P-Spline Regression Models Estimation

Version 1.0-1

Date 2023-09-15

Imports stats, MASS

Suggests SpATS

Maintainer Maria Xose Rodriguez-Alvarez <mxrodriguez@uvigo.gal>

Description Generalised additive P-spline regression models estimation using the separation of overlapping precision matrices (SOP) method. Estimation is based on the equivalence between P-splines and linear mixed models, and variance/smoothing parameters are estimated based on restricted maximum likelihood (REML). The package enables users to estimate P-spline models with overlapping penalties. Based on the work described in Rodriguez-Alvarez et al. (2015) <doi:10.1007/s11222-014-9464-2>; Rodriguez-Alvarez et al. (2019) <doi:10.1007/s11222-018-9818-2>, and Eilers and Marx (1996) <doi:10.1214/ss/1038425655>.

License GPL-2

NeedsCompilation no

Author Maria Xose Rodriguez-Alvarez [aut, cre] (<https://orcid.org/0000-0002-1329-9238>),
Manuel Oviedo de la Fuente [aut],
Maria Durban [ctb],
Paul H.C. Eilers [ctb],
Dae-Jin Lee [ctb],
Mikis Stasinopoulos [ctb]

Repository CRAN

Date/Publication 2023-09-15 21:52:12 UTC

Contents

`SOP-package` *Generalised Additive P-Spline Regression Models Estimation*

Description

Generalised additive P-spline regression models estimation using the separation of overlapping precision matrices (SOP) method. Estimation is based on the equivalence between P-splines and linear mixed models, and variance/smoothing parameters are estimated based on restricted maximum likelihood (REML). The package enables users to estimate P-spline models with overlapping penalties. Based on the work described in Rodriguez-Alvarez et al. (2015) <doi:10.1007/s11222-014-9464-2>; Rodriguez-Alvarez et al. (2019) <doi:10.1007/s11222-018-9818-2>, and Eilers and Marx (1996) <doi:10.1214/ss/1038425655>.

Details

Index of help topics:

```
SOP-package             Generalised Additive P-Spline Regression Models
                        Estimation
ad                      Adaptive smooth terms in a SOP model formula
f                       Defining smooth terms in SOP formulae
plot.sop                Default SOP plotting
predict.sop             Prediction from a fitted SOP model
print.sop               Print method for sop objects
rae                     Defining random effects in SOP formula
sop                     Estimation of generalised additive P-spline
                        regression models with overlapping penalties.
sop.control             Function for controlling SOP fitting
sop.fit                 Fitting generalised linear mixed models with
                        overlapping precision matrices.
summary.sop             Summary method for a fitted SOP model.
```

This package incorporates the function `sop()` which enables users to estimate multidimensional generalised P-spline regression models with overlapping penalties. For a complete list of functions use `library(help = SOP)`.

Author(s)

NA

Maintainer: Maria Xose Rodriguez-Alvarez <mxrodriguez@uvigo.gal>

References

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, **11 (2)**, 89–121.

Rodriguez-Alvarez, M.X., Lee, D. J., Kneib, T., Durban, M., and Eilers, P. (2015). Fast smoothing parameter separation in multidimensional generalized P-splines: the SAP algorithm. *Statistics and Computing*, **25 (5)**, 941–957.

Rodriguez-Alvarez, M.X., Durban, M., Lee, D. J. and Eilers, P. (2019). On the estimation of variance parameters in non-standard generalised linear mixed models: application to penalised smoothing. *Statistics and Computing*, **29 (3)**, 483–500.

`ad` *Adaptive smooth terms in a SOP model formula*

Description

Auxiliary function used to define adaptive smooth terms in a `sop` model formula. The function does not do any fitting but is used as part of a `sop()` model formula.

Usage

```
ad(..., nseg = 10, pord = 2, degree = 3, nseg.sp = 5, degree.sp = 3)
```

Arguments

`...`	the x-variable (continuous) to be used for adaptive smoothing. Currently, only one dimensional adaptive smoothers are allowed.
`nseg`	the number of segments for the B-spline basis used to represent the smooth term. The default value is 10.
`pord`	penalty order. The defauly value is 2 (second order penalty).
`degree`	the order of the polynomial for the B-spline basis for this term. The default value is 3 (cubic B-splines).
`nseg.sp`	the number of segments for the B-spline basis used to ‘smooth’ the smoothing parameters. The default value is 5.
`degree.sp`	the order of the polynomial for the B-spline basis used for smoothing of the smoothing parameters. The default value is 3 (cubic B-splines).

Details

The function `ad()` can be use to fit an adaptive smooth function of `x`. An ‘adaptive’ smooth function is one in which the smoothing parameter is allowed to vary over the range of the explanatory variable `x`. Details can be found in Rodriguez-Alvarez *at. al* (2019).

Value

The function is interpreted in the formula of a sop model and creates the right framework for fitting the adaptive smoother. List containing the following objects:

`vars`	name of the covariates involved in the adaptive smooth term.
`nseg`	the number of segments for the B-spline basis.
`pord`	the penalty order.
`degree`	the order of the polynomial for the B-Spline basis for this term.
`nseg.sp`	the number of segments for the B-spline basis used to 'smooth' the smoothing parameters.
`degree.sp`	the order of the polynomial for the B-spline basis used for smoothing of the smoothing parameters.
`dim`	The dimension of the smoother - i.e. the number of covariates that it is a function of.
`label`	labels terms.

References

Rodriguez-Alvarez, M.X., Durban, M., Lee, D. J. and Eilers, P. (2019). On the estimation of variance parameters in non-standard generalised linear mixed models: application to penalised smoothing. *Statistics and Computing*, **29 (3)**, 483–500.

See Also

f, rae, sop

Examples

```
library(SOP)
# Simulate the data
set.seed(123)
n <- 1000
x <- runif(n, 0.0001, 1)
doppler.function <- function(x) sin(4 / (x + 0.1)) + 1.5
mu <- doppler.function(x)
sigma <- 0.2
y <- mu + sigma*rnorm(n)
dat <- data.frame(x = x, y = y)

# Fit the models
# With addaptive smoothing
m0 <- sop(formula = y ~ ad(x, nseg = 197, nseg.sp = 17), data = dat,
         control =  list(trace = FALSE, epsilon = 1e-03))

# Without addaptive smoothing
m1 <- sop(formula = y ~ f(x, nseg = 197), data = dat,
         control =  list(trace = FALSE, epsilon = 1e-03))

# Plot results
```

```
plot(y ~ x, data = dat)
ox <- order(dat$x)
lines(fitted(m0)[ox] ~ dat$x[ox], col = 2, lwd = 2)
lines(fitted(m1)[ox] ~ dat$x[ox], col = 4, lwd = 2)
legend("topright", c("Theoretical", "Adaptive", "Non Adaptive"),
    col = c(1,2,4), lty = 1, lwd = 2, bty = "n")
```

`f` *Defining smooth terms in SOP formulae*

Description

Auxiliary function used to define smooth terms within `sop()` model formulae. The function does not evaluate the smooth - it exists purely to help set up a model using P-spline based smoothers.

Usage

```
f(..., nseg = 10, pord = 2 , degree = 3)
```

Arguments

`...`	a list of up to three variables to construct the smooth term.
`nseg`	the number of segments for the (marginal) B-spline bases used to represent the smooth term. Numerical vector of length equal to the number of covariates. Atomic values are also valid, being recycled. The default value is 10.
`pord`	penalty order. Numerical vector of length equal to the number of covariates. Atomic values are also valid, being recycled. The default value is 2 (second-order penalty).
`degree`	the order of the polynomial for the (marginal) B-spline bases for this term. Numerical vector of length equal to the number of covariates. Atomic values are also valid, being recycled. The default value is 3 (cubic B-splines).

Details

The functions `f()` is designed to represent either a one dimensional smooth functions for main effects of an continuous explanatory variable or two or three dimensional smooth functions representing two way and three way interactions of continuous variables. By default, the values of the arguments `nseg`, `pord` and `degree` are repeated to the length of the explanatory covariates. The two and three dimensional smooth terms are constructed using the tensor-product of marginal (one-dimensional) B-spline bases and anisotropic penalties are considered.

Value

The function is interpreted in the formula of a `sop` model and creates the right framework for fitting the smoother. List containing the following elements:

`vars`	names of the covariates involved in the smooth term.

`nseg`	the number of segments for the (marginal) B-spline basis for each covariate.
`pord`	the penalty order (numerical vector of length equal to the number of covariates).
`degree`	the order of the polynomial for the (marginal) B-Spline bases for this term (numerical vector of length equal to the number of covariates).
`dim`	The dimension of the smoother - i.e. the number of covariates that it is a function of.
`label`	labels terms.

See Also

`ad`, `rae`, `sop`

Examples

```
library(SOP)
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.5
x <- runif(n)
f0 <- function(x) 2*sin(pi*x)
f <- f0(x)
y <- f + rnorm(n, 0, sigma)
dat <- data.frame(x = x, y = y)

# Fit the model
m0 <- sop(formula = y ~ f(x, nseg = 10), data = dat)
summary(m0)

# Plot results
plot(y ~ x, data = dat)
ox <- order(dat$x)
lines(f[ox] ~ dat$x[ox], lwd = 2)
lines(fitted(m0)[ox] ~ dat$x[ox], col = "red", lwd = 2)
```

`plot.sop` *Default SOP plotting*

Description

Takes a fitted sop object produced by `sop()` and plots the component smooth functions that make it up, on the scale of the linear predictor.

Usage

```
## S3 method for class 'sop'
plot(x, rug = TRUE, pages = 0, select = NULL, grid, ...)
```

Arguments

`x`	a fitted `sop` object as produced by `sop()`.
`rug`	when TRUE (default) then the covariate to which the plot applies is displayed as a rug plot at the foot of each plot of a 1-d smooth. Setting to `FALSE` will speed up plotting for large datasets.
`pages`	(default 0) the number of pages over which to spread the output. For example, if `pages=1` then all terms will be plotted on one page with the layout performed automatically. Set to 0 to have the routine leave all graphics settings as they are.
`select`	Allows the plot for a single model smooth term to be selected for printing. e.g. if you just want the plot for the second smooth term set `select = 2`.
`grid`	number of covariate values used for each 1-d plot - for a nice smooth plot this needs to be several times the estimated degrees of freedom for the smooth. Default value 100.
`...`	other graphics parameters to pass on to plotting commands. See details for smooth plot specific options.

Details

Produces default plot showing the smooth and random components of a fitted SOP.

For smooth terms `plot.sop` actually calls plot method functions depending on the dimension of the smooth function.

For plots of smooths in one dimension, the x axis of each plot is labelled with the covariate name, while the y axis is labelled '`f(cov),edf`' where `cov` is the covariate name, and `edf` the estimated degrees of freedom of the smooth.

Several smooth plots methods using `image` will accept a `colors` argument, which can be anything documented in `topo.colors` (in which case something like `colors=rainbow(50)` is appropriate), or the `grey` function (in which case something like `colors=grey(0:50/50)` is needed).

Value

The function main purpose is to generate plots. It also (silently) returns a list of the data used to produce the plots for the smooth terms. This function is inspired by the plot.gam function of the same name described in mgcv package (but is not a clone).

See Also

`sop`, `predict.sop`

Examples

```
library(SOP)
## Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.5
x <- runif(n)
f0 <- function(x) 2*sin(pi*x)
```

```
f <- f0(x)
y <- f + rnorm(n, 0, sigma)
dat <- data.frame(x = x, y = y)

# Fit the model
m0 <- sop(formula = y ~ f(x, nseg = 10), data = dat)
summary(m0)

# Plot results
plot(m0)

## An example of use of SOP package with tensor product B-splines in 2D
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.1
x1 <- runif(n, -1, 1)
x2 <- runif(n, -1, 1)
f0 <- function(x1, x2) cos(2*pi*sqrt((x1 - 0.5)^2 + (x2 - 0.5)^2))
f <- f0(x1, x2)
y <- f + rnorm(n, 0, sigma)

dat <- data.frame(x1 = x1, x2 = x2, y = y)

m0 <- sop(formula = y ~ f(x1, x2, nseg = 10), data = dat,
        control = list(trace = FALSE))

summary(m0)
plot(m0, col = topo.colors(100))

plot(m0, col = grey(0:100/100))

aux <- plot(m0)
names(aux)
```

`predict.sop` *Prediction from a fitted SOP model*

Description

The function takes a fitted sop object and produces predictions for the original data if the argument `newdata` is not set or predictions for new data if `newdata` is specified. Predictions can be accompanied by standard errors, based on the Bayesian posterior distribution of the model coefficients.

Usage

```
## S3 method for class 'sop'
predict(object, newdata, type = c("response", "link", "terms"),
      se.fit = FALSE, ...)
```

Arguments

`object`	a fitted sop object as produced by `sop()`.
`newdata`	a data frame containing the values of the model covariates at which predictions are required. If this is not provided then predictions corresponding to the original data are returned. If the data frame `newdata` is provided then it should contain all the variables needed for prediction: a warning is generated if not. If `newdata` contains a variable `offset`, it is included into the predictions when `type = "link"` and `type = "response"`.
`type`	When this has the value `"link"` the linear predictor fitted values or predictions (possibly with associated standard errors) are returned. When `type = "terms"` each component of the linear predictor is returned separately (possibly with approximate standard errors): this includes parametric model components, followed by each smooth component, but excludes any offset and any intercept. When `type = "response"` (default) fitted values or predictions on the scale of the response are returned (possibly with approximate standard errors).
`se.fit`	when this is TRUE (not default) standard error estimates are returned for each prediction.
`...`	other arguments. Not yet implemented.

Value

A vector/matrix (or list, with elements `fit` and `se.fit`, is `se = TRUE`) equal to:

`"link"`	a vector of linear predictor values.
`"response"`	a vector of linear predictor values on the scale of the response.
`"terms"`	a matrix with a column per term, and may have an attribute "constant".

See Also

`sop`, `plot.sop`

Examples

```
library(SOP)
## Example training/set
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.5
x <- runif(n)
f0 <- function(x)2*sin(pi*x)
f <- f0(x)
y <- f + rnorm(n, 0, sigma)
da <- data.frame(x = x, y = y)# all data
rand <-  sample(2, 610, replace=TRUE, prob=c(0.6,0.4))
traindata <- da[rand==1,] # training data
valdata <- da[rand==2,] # validation data
plot(y ~ x, data = traindata, pch = 20, col = gray(.7))
```

```
points(y ~ x, data = valdata, pch = 20, col = gray(.2))

# Fit the model in the training data
m0 <- sop(formula = y ~ f(x, nseg = 10), data = traindata)
lines(fitted(m0)[order(traindata$x)]~traindata$x[order(traindata$x)],
       col="red", lwd=2)

# Predict and plot in the data used for the fit
po <- predict(m0)
plot(y ~ x, data = traindata, pch = 20, col = gray(.7))
lines(po[order(traindata$x)] ~ traindata$x[order(traindata$x)],
      col="red", lwd=2)

# Predict and plot in new data
pn <- predict(m0, newdata = valdata)
plot(y ~ x, data = traindata, pch = 20, col = gray(.7))
lines(pn[order(valdata$x)] ~ valdata$x[order(valdata$x)], col = "yellow", lwd = 2)

# Example Gamma distribution
# Simulate the data
set.seed(123)
n <- 1000
alpha <- 0.75
x0 <- runif(n)
x1 <- x0*alpha + (1-alpha)*runif(n)
x2 <- runif(n)
x3 <- x2*alpha + (1-alpha)*runif(n)
x4 <- runif(n)
x5 <- runif(n)

f0 <- function(x)2*sin(pi*x)
f1 <- function(x)exp(2*x)
f2 <- function(x) 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10

f <- f0(x0) + f1(x1) + f2(x2)
y <- rgamma(f,exp(f/4),scale=1.2)

df <- data.frame(y = y, x0 = x0, x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = x5)

# Fit the model
m1 <- sop(formula = y ~ f(x0, nseg = 17) + f(x1, nseg = 17) +
      f(x2, nseg = 17) + f(x3, nseg = 17) +
      f(x4, nseg = 17) + f(x5, nseg = 17),
      family = Gamma(link = log), data = df)
summary(m1)

# Predict in a new dataframe
x <- seq(max(c(min(x1),min(x3))), min(c(max(x1),max(x3))), l = 100)
df.p <- data.frame(x0 = x, x1 = x, x2 = x, x3 = x, x4 = x, x5 = x)
p <- predict(m1, type = "terms", newdata = df.p)
colnames(p)

# Plot the different smooth terms
```

```
op <- par(mfrow = c(2,3))
plot(m1, select = 1)
lines(x, p[,1], col = "red")
plot(m1, select = 2)
lines(x, p[,2], col = "red")
plot(m1, select = 3)
lines(x, p[,3], col = "red")
plot(m1, select = 4)
lines(x, p[,4], col = "red")
plot(m1, select = 5)
lines(x, p[,5], col = "red")
plot(m1, select = 6)
lines(x, p[,6], col = "red")
par(op)
```

`print.sop` *Print method for sop objects*

Description

Print method for sop objects

Usage

```
## S3 method for class 'sop'
print(x, ...)
```

Arguments

`x`	an object of class `sop` as produced by `sop()`
`...`	further arguments passed to or from other methods. Not yet implemented.

Value

Prints some summary statistics of the fitted model.

See Also

`sop`

Examples

```
library(SOP)
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.5
x <- runif(n)
f0 <- function(x) 2*sin(pi*x)
```

```
f <- f0(x)
y <- f + rnorm(n, 0, sigma)
dat <- data.frame(x = x, y = y)

# Fit the model
m0 <- sop(formula = y ~ f(x, nseg = 10), data = dat)
m0
```

---

rae *Defining random effects in SOP formula*

Description

Auxiliary function used to define random effects terms in a sop model formula.

Usage

```
rae(x)
```

Arguments

x	the x-variable (factor) that defines the random effects term.

Details

The functions is designed to represent random effects in SOP formulae.

Value

The function is interpreted in the formula of a sop model and creates the right framework for fitting the random effect. List containing the following elements:

x	name of the covariate involved.

See Also

f, ad, sop

Examples

```
library(SOP)
require(SpATS)
## An example of use of SOP package for the analysis of field trials experiments.
## Taken from the SpATS package.
data(wheatdata)

# Create factor variable for row and columns
wheatdata$R <- as.factor(wheatdata$row)
wheatdata$C <- as.factor(wheatdata$col)
```

```
# package SOP
m0 <- sop(formula = yield ~ colcode + rowcode +
  f(col, row, nseg = c(10, 10)) +
  rae(geno) + rae(R) + rae(C), data = wheatdata)
summary(m0)
plot(m0, col = topo.colors(100))
```

---

sop *Estimation of generalised additive P-spline regression models with overlapping penalties.*

Description

The function `sop()` fits generalised additive regression models. For the smooth terms, it uses P-splines (Eilers and Marx, 1996) and it can cope with one, two and three dimensional smooth terms. The innovation of the function is that smoothing/variance parameters are estimated on the basis of the SOP method; see Rodriguez-Alvarez *et al.* (2015) and Rodriguez-Alvarez *et al.* (2019) for details. This speeds up the fit.

Usage

```
sop(formula, data = list(),  family = gaussian(), weights = NULL, offset = NULL,
    control = sop.control(), fit = TRUE)
```

Arguments

`formula`	a sop formula. This is exactly like the formula for a GLM except that (1) P-splines in one, two and three dimensions (`f`), (2)spatially adaptive P-splines in 1 dimension (`ad`); and (3) random effects (`rae`) can be added to the right hand side of the formula.
`data`	a data frame containing the model response variable and covariates required by the formula.
`family`	object of class `family` specifying the distribution and link function.
`weights`	prior weights on the contribution of the data to the log likelihood. Note that a weight of 2, for example, is equivalent to having made exactly the same observation twice. If you want to reweight the contributions of each datum without changing the overall magnitude of the log likelihood, then you should normalize the weights e.g. `weights <- weights/mean(weights)`). If NULL (default), the weights are considered to be one.
`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of observations.
`control`	a list of control values to replace the default values returned by the function `sop.control`.
`fit`	logical. If TRUE, the model is fitted.

Details

The `sop()` can be used to fit generalised additive models. It works similarly to the function `gam()` of the package **mgcv**. The function `sop()` uses P-splines (Eilers and Marx, 1996), one of the option on `gam()`. Estimation is based on the equivalence between P-splines and linear mixed models, and variance/smoothing parameters are estimated based on restricted maximum likelihood (REML) using the separation of overlapping precision matrices (SOP) method described in Rodriguez-Alvarez *et al.* (2015) and Rodriguez-Alvarez *et al.* (2019). The function `sop()` can be seen as a faster alternative to `gam()` for some data sets.

Value

An object of class 'sop'. It is a list containing the following objects:

`b.fixed`	the estimated fixed effect coefficients (present if `fit = TRUE`).
`b.random`	the predicted random effect coefficients (present if `fit = TRUE`).
`fitted.values`	the fitted values (present if `fit = TRUE`).
`linear.predictor`	the values of the linear predictor (present if `fit = TRUE`).
`residuals`	the (deviance) residuals (present if `fit = TRUE`).
`X`	the fixed effect design matrix.
`Z`	the random effect design matrix.
`G`	a list containing information about the precision/penalty matrices (one for each smoothing/variance parameter in the model).
`y`	the response
`weights`	the prior weights.
`family`	the distribution family.
`out`	a list with i) `tol.ol` tolerance parameter (outer loop); ii) `it.ol` number of iteration (outer loop); iii) `tol.il` tolerance parameter (inner loop); `it.il` (number of iteration (inner loop)), iv) `vc` variance components estimates, v) `edf` effective degrees of freedom (present if `fit = TRUE`).
`deviance`	the deviance (present if `fit = TRUE`).
`null.deviance`	the null deviance (present if `fit = TRUE`).
`Vp`	Bayesian posterior covariance matrix for the coefficients (present if `fit = TRUE`).
`call`	the function call.
`data`	the data.
`formula`	the model formula.
`lin`	a list containing information about the parametric/linear.
`random`	a list containing information about the random effects.
`f`	a list containing information about the smoothers.
`na.action`	vector with the observations (position) deleted due to missingness.
`names.terms`	the terms used in the formula.
`model.terms`	the explanatory variables.
`nterms`	the number of linear, random and smooth terms in the formula.

References

Eilers, P.H.C. and Marx, B.D. (1996). Flexible smoothing with B-splines and penalties. *Statistical Science*, **11 (2)**, 89–121.

Rodriguez-Alvarez, M.X., Lee, D. J., Kneib, T., Durban, M., and Eilers, P. (2015). Fast smoothing parameter separation in multidimensional generalized P-splines: the SAP algorithm. *Statistics and Computing*, **25 (5)**, 941–957.

Rodriguez-Alvarez, M.X., Durban, M., Lee, D. J. and Eilers, P. (2019). On the estimation of variance parameters in non-standard generalised linear mixed models: application to penalised smoothing. *Statistics and Computing*, **29 (3)**, 483–500.

Examples

```
library(SOP)
## An example of use of SOP package with tensor product B-splines in 2D
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.1
x1 <- runif(n, -1, 1)
x2 <- runif(n, -1, 1)
f0 <- function(x1, x2) cos(2*pi*sqrt((x1 - 0.5)^2 + (x2 - 0.5)^2))
f <- f0(x1, x2)
y <- f + rnorm(n, 0, sigma)

dat <- data.frame(x1 = x1, x2 = x2, y = y)

# Theoretical surface
np <- 50
x1p <- seq(-1, 1, length = np)
x2p <- seq(-1, 1, length = np)
fp <- cos(2 * pi * sqrt(outer((x1p - 0.5) ^ 2, (x2p - 0.5) ^ 2, '+')))

image(x1p, x2p, matrix(fp, np, np), main = 'f(x1,x2) - Theor',
     col = topo.colors(100))

# Fit the model
m0 <- sop(formula = y ~ f(x1, x2, nseg = 10), data = dat,
        control = list(trace = FALSE))

summary(m0)
plot(m0, col = topo.colors(100))

## An example of use of SOP package with several smooth terms and Gamma distribution
# Simulate the data
set.seed(123)
n <- 1000
alpha <- 0.75
x0 <- runif(n)
x1 <- x0*alpha + (1-alpha)*runif(n)
x2 <- runif(n)
x3 <- x2*alpha + (1-alpha)*runif(n)
```

```
x4 <- runif(n)
x5 <- runif(n)

f0 <- function(x)2*sin(pi*x)
f1 <- function(x)exp(2*x)
f2 <- function(x) 0.2*x^11*(10*(1-x))^6+10*(10*x)^3*(1-x)^10

f <- f0(x0) + f1(x1) + f2(x2)
y <- rgamma(f,exp(f/4),scale=1.2)

df <- data.frame(y = y, x0 = x0, x1 = x1, x2 = x2, x3 = x3, x4 = x4, x5 = x5)

# Fit the model
m1 <- sop(formula = y ~ f(x0, nseg = 17) +
                             f(x1, nseg = 17) +
                             f(x2, nseg = 17) +
                             f(x3, nseg = 17) +
                             f(x4, nseg = 17) +
                             f(x5, nseg = 17), family = Gamma(link = log), data = df)
summary(m1)
plot(m1)

## An example of use of SOP package for the analysis of field trials experiments.
## Taken from the SpATS package.
require(SpATS)
data(wheatdata)

# Create factor variable for row and columns
wheatdata$R <- as.factor(wheatdata$row)
wheatdata$C <- as.factor(wheatdata$col)

# package SOP
m2 <- sop(formula = yield ~ colcode + rowcode +
  f(col, row, nseg = c(10, 10)) +
  rae(geno) + rae(R) + rae(C), data = wheatdata)
summary(m2)
plot(m2, col = topo.colors(100), pages = 1)

# Package SpATS: more adequate for this analysis.
# SpATS has been explicitly developed for the analysis field trials experiments.
m3 <- SpATS(response = "yield",
    spatial = ~ SAP(col, row, nseg = c(10,10), degree = 3, pord = 2, center = TRUE),
    genotype = "geno",
    genotype.as.random = TRUE,
    fixed = ~ colcode + rowcode, random = ~ R + C, data = wheatdata,
    control =  list(tolerance = 1e-06))
summary(m3)
plot(m3)
```

sop.control *Function for controlling SOP fitting*

Description

The function controls some of the fitting parameters of `sop`. Typically only used when calling `sop()`.

Usage

```
sop.control(maxit = 200, epsilon = 1e-6, trace = FALSE)
```

Arguments

`maxit`	numerical value indicating the maximum number of iterations. Default set to 200 (see Details).
`epsilon`	numerical value indicating the tolerance for the convergence criterion. Default set to 1e-6 (see Details).
`trace`	logical indicating if output should be produced for each iteration.

Details

For Gaussian response variables, the implemented algorithm is an iterative procedure, with the fixed and random effects as well as the variance components being updated at each iteration. To check the convergence of this iterative procedure, the (REML) deviance is monitored. For non-Gaussian response variables, estimation is based on Penalized Quasi-likelihood (PQL) methods. Here, the algorithm is a two-loop algorithm: the outer loop corresponds to the Fisher-Scoring algorithm (monitored on the basis of the change in the linear predictor between consecutive iterations), and the inner loop corresponds to that described for the Gaussian case.

Value

A list with the arguments as components.

References

Rodriguez-Alvarez, M.X., Lee, D. J., Kneib, T., Durban, M., and Eilers, P. (2015). Fast smoothing parameter separation in multidimensional generalized P-splines: the SAP algorithm. *Statistics and Computing*, **25 (5)**, 941–957.

Rodriguez-Alvarez, M.X., Durban, M., Lee, D. J. and Eilers, P. (2019). On the estimation of variance parameters in non-standard generalised linear mixed models: application to penalised smoothing. *Statistics and Computing*, **29 (3)**, 483–500.

See Also

`sop`.

Examples

```
library(SOP)
# Simulate the data
set.seed(123)
n <- 1000
```

```
sigma <- 0.5
x <- runif(n)
f0 <- function(x) 2*sin(pi*x)
f <- f0(x)
y <- f + rnorm(n, 0, sigma)
dat <- data.frame(x = x, y = y)

# Fit the model
m0 <- sop(formula = y ~ f(x, nseg = 10), data = dat, control = list(trace = FALSE))
summary(m0)
```

`sop.fit` *Fitting generalised linear mixed models with overlapping precision matrices.*

Description

This is an internal function of package SOP. It is used to fit SOP models by specifying the design matrices for the fixed and random effects as well as the precision matrices for each variance component in the model.

Usage

```
sop.fit(y, X, Z, weights = NULL, G = NULL, vcstart = NULL,
  etastart = NULL, mustart = NULL, offset = NULL,
  family = gaussian(), control = sop.control())
```

Arguments

`y`	vector of observations of length n.
`X`	design matrix for the fixed effects (dimension n x p).
`Z`	design matrix for the random effects (of dimension n x q).
`weights`	an optional vector of 'prior weights' to be used in the fitting process. If NULL (default), the weights are considered to be one.
`G`	a list with the diagonal elements of the precision matrices for each variance component in the model. Each element of the list is a vector of the same length as the number of columns in Z (i.e. q). The vector can be padded out with zeroes to indicate random coefficient not 'affected' by the variance component (see details).
`vcstart`	optional numeric vector. Initial values for the variance components (including the error variance as the first element of the vector). If `NULL`, all variance components are initialised to one.
`etastart`	initial values for the linear predictor.
`mustart`	initial values for the expected response.

`offset`	this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of observations.
`family`	object of class `family` specifying the distribution and link function.
`control`	a list of control values to replace the default values returned by the function `sop.control`.

Details

`sop.fit` is the workhorse function: it is typically not normally called directly but can be more efficient where the response vector 'y', design matrixs 'X' and 'Z', and precision matrices 'G' have already been calculated. Currently, the funcion only allows for diagonal precision matrices (possibly overlappping).

Value

A list containing the following objects:

`b.fixed`	the estimated fixed effect coefficients.
`b.random`	the predicted random effect coefficients.
`residuals`	the (deviance) residuals.
`fitted.values`	the fitted values.
`linear.predictor`	the values of the linear predictor.
`X`	the fixed effect design matrix.
`Z`	the random effect design matrix.
`y`	the response.
`weights`	the prior weights.
`family`	the distribution family.
`out`	a list with i) `tol.ol` tolerance parameter (outer loop); ii) `it.ol` number of iteration (outer loop); iii) `tol.il` tolerance parameter (inner loop); `it.il` (number of iteration (inner loop)), iv) `vc` variance components estimates, v) `edf` effective degrees of freedom.
`deviance`	the deviance.
`null.deviance`	the null deviance.
`Vp`	Bayesian posterior covariance matrix for the coefficients.

References

Rodriguez-Alvarez, M.X., Lee, D. J., Kneib, T., Durban, M., and Eilers, P. (2015). Fast smoothing parameter separation in multidimensional generalized P-splines: the SAP algorithm. *Statistics and Computing*, **25 (5)**, 941–957.

Rodriguez-Alvarez, M.X., Durban, M., Lee, D. J. and Eilers, P. (2019). On the estimation of variance parameters in non-standard generalised linear mixed models: application to penalised smoothing. *Statistics and Computing*, **29 (3)**, 483–500.

Examples

```
library(SOP)
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.1
x1 <- runif(n, -1, 1)
x2 <- runif(n, -1, 1)
f0 <- function(x1, x2) cos(2*pi*sqrt((x1 - 0.5)^2 + (x2 - 0.5)^2))
f <- f0(x1, x2)
y <- f + rnorm(n, 0, sigma)
dat <- data.frame(x1 = x1, x2 = x2, y = y)

# Save but not fit the model
m0_nfit <- sop(formula = y ~ f(x1, x2, nseg = 10), data = dat,
     fit = FALSE)

# Now fit using sop.fit()
m0 <- sop.fit(X = m0_nfit$X, Z = m0_nfit$Z, G = m0_nfit$G,
y = m0_nfit$y, weights = m0_nfit$weights,
   control = list(trace = FALSE))

names(m0)
```

`summary.sop` *Summary method for a fitted SOP model.*

Description

Summary method for a fitted SOP model.

Usage

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

Arguments

`object`	an object of class `sop` as produced by `sop()`.
`...`	further arguments passed to or from other methods. Not yet implemented.

Value

The function `summary.sop` computes and returns a list of summary statistics of the fitted model given in `object`, using the components (list elements) `"call"` and `"terms"` from its argument, plus

`call`	the matched call.
`b.random`	a vector with the predicted random effects coefficients.

`b.fixed`	a vector with the estimated fixed effects coefficients.
`r.sq.adj`	the (adjusted) R^2, i.e., 'fraction of variance explained by the model', $$R^2 = 1 - \frac{\sum_i R_i^2/(n - df)}{\sum_i (y_i - y^*)^2/(n-1)},$$ where $R_i = w_i(y_i - \mu_i)$ and y^* is the (weighted) mean of y_i.
`deviance`	the deviance.
`null.deviance`	the null deviance.
`dev.expl`	proportion of the null deviance explained by the model.
`n`	number of data.
`iter`	number of iterations.
`residual.df`	residual degrees of freedom.
`edf`	a vector with the estimated degrees of freedom for the (smooth and random) model terms.
`formula`	the model formula.
`family`	the family used.
`na.action`	vector with the observations (position) deleted due to missingness.

See Also

`sop`, `summary`

Examples

```
library(SOP)
# Simulate the data
set.seed(123)
n <- 1000
sigma <- 0.5
x <- runif(n)
f0 <- function(x) 2*sin(pi*x)
f <- f0(x)
y <- f + rnorm(n, 0, sigma)
dat <- data.frame(x = x, y = y)

# Fit the model
m0 <- sop(formula = y ~ f(x, nseg = 10), data = dat)
summary(m0)
```

Index