Package 'psme'

May 9, 2026

Version 1.0.0

Date 2025-10-06

Title Penalized Splines Mixed-Effects Models

Maintainer Zheyuan Li <zheyuan.li@bath.edu>

Depends R (>= 4.0.0)

Imports stats, Matrix, methods, mgcv, lme4

Description

Fit penalized splines mixed-effects models (a special case of additive models) for large longitudinal datasets. The package includes a psme() function that (1) relies on package 'mgcv' for constructing population and subject smooth functions as penalized splines, (2) transforms the constructed additive model to a linear mixed-effects model, (3) exploits package 'lme4' for model estimation and (4) backtransforms the estimated linear mixed-effects model to the additive model for interpretation and visualization. See Pedersen et al. (2019) <doi:10.7717/peerj.6876> and Bates et al. (2015) <doi:10.18637/jss.v067.i01> for an introduction. Unlike the gamm() function in 'mgcv', the psme() function is fast and memory-efficient, able to handle datasets with millions of observations.

License GPL-3

NeedsCompilation no

URL https://github.com/ZheyuanLi/psme

Author Zheyuan Li [aut, cre] (ORCID: <https://orcid.org/0000-0002-7434-5947>)

Repository CRAN

Date/Publication 2025-10-09 12:00:06 UTC

Contents

`EvalSmooth` *Evaluate a fitted smooth term in a fitted model*

Description

A helper function to evaluate the fitted population and subject smooth functions in a fitted penalized splines mixed-effects model.

Usage

```
EvalSmooth(mgcv.smooth, new.x)
```

Arguments

`mgcv.smooth`	A single smooth term taken from the `smooth` list of the fitted penalized splines mixed-effects model.
`new.x`	A numeric vector of new covariate values at which to evaluate the smooth term.

Value

These are two common cases of population smooth functions. For a single smooth function like `s(x)`, the function returns a vector of predicted values. For a factor 'by' smooth term like `s(x, by = f)`, the function returns a matrix of predicted values, where each column corresponds to a factor level.

For subject smooth functions specified as a factor-smooth interaction term `s(x, f, bs = "fs")`, the function returns a matrix of predicted values, where each column corresponds to an individual subject.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

See Also

`psme`

Examples

```
## see examples for the psme() function
```

GetExtrema *Identify all the local extrema of a curve with a grid search*

Description

Approximately identify all the local extrema of a 1-dimensional curve $y = f(x)$, by finding all the local minima and all the local maxima in a sequence of (x, y) values, discretely observed or sampled from the curve.

Usage

```
GetExtrema(x, y)
```

Arguments

x	A numeric vector of x-coordinates for a curve.
y	A numeric vector of y-coordinates for a curve.

Value

A list with four elements:

min.x	The x-coordinates of the local minima.
min.y	The y-coordinates of the local minima.
max.x	The x-coordinates of the local maxima.
max.y	The y-coordinates of the local maxima.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

See Also

EvalSmooth

Examples

```
x <- seq(0, 2 * pi, length.out = 100)
y <- sin(x)
ex <- GetExtrema(x, y)
plot(x, y, type = "l")
points(ex$min.x, ex$min.y, pch = 19, col = 2)
points(ex$max.x, ex$max.y, pch = 19, col = 3)
```

psme *Fit penalized splines mixed-effects models*

Description

Fit penalized splines mixed-effects models by leveraging the known equivalence between penalized splines and mixed-effects models. `psme` reparameterizes each penalized spline term specified in a *mgcv*-style model formula as linear mixed-effects model terms, and fits such equivalent model using *lme4* as the computational engine.

Usage

```
psme(mgcv.form, data, knots = NULL)
```

Arguments

`mgcv.form`	a *mgcv*-style model formula that specify a penalized splines mixed-effects model in the fashion of penalized splines additive models.
`data`	a data frame with all variables in the formula, and any rows with NA must be mannually removed.
`knots`	an optional named list providing knots.

Details

A penalized splines mixed-effects model is a special case of additive models, most suitable for modelling longitudinal data. Such model typically contains some population smooth functions and a group of subject smooth functions, both nonlinear in a variable (for example, time). The function relies on *mgcv* for constructing smooth functions as penalized splines, and users should specify these terms via a *mgcv*-style model formula. Examples of population smooth functions include a single smooth function like `s(x)` and a factor 'by' smooth term like `s(x, by = f)`, where the smooth function in `x` changes with the levels of a factor variable `f` (for example, gender). Subject smooth functions are set up as a factor-smooth interaction term like `s(x, f, bs = "fs")`. The model may also involve smooth functions of other covariate variables constructed using `s`, `te` and `ti`, as well as any parametric terms that are legitimate in a `lm` formula. In addition, users can customize the class and basis dimension of each penalized spline, as they can when using *mgcv*.

To fit a penalized splines mixed-effects model, the function transforms each penalized spline into a combination of fixed and random effects, producing an equivalent linear mixed-effects model. This model is then estimated using *lme4*'s low-level functions. Finally, the function backtransforms the estimated fixed-effects and predicted random-effects to the basis coefficients of the original penalized splines, so that the users can extract and plot the estimated smooth functions. In particular, the transformation from a penalized splines mixed-effects model to a linear mixed-effects model is performed with great caution to preserve the sparsity of the design matrices. As a result, the function is able to handle large longitudinal datasets with millions of observations.

Value

a list containing:

`pform`	The parametric component of the model formula.
`pcoef`	The estimated coefficients of the parametric component of the model.
`smooth`	A list of constructed and estimated smooth functions corresponding to the smooth terms in the model formula.
`lme4.fit`	The raw output of *lme4*'s low-level functions.

Author(s)

Zheyuan Li <zheyuan.li@bath.edu>

See Also

gam, gamm

Examples

```
library(psme)

## simulate a toy dataset of 50 subjects, each with a random quadratic trajectory
n.subjects <- 50
y <- x <- vector("list", n.subjects)
## a subject has 5 to 10 random observations
n <- sample(5:10, size = n.subjects, replace = TRUE)
a <- rnorm(n.subjects, mean = 1, sd = 0.2)
b <- rnorm(n.subjects, mean = 0, sd = 0.3)
c <- rnorm(n.subjects, mean = 0, sd = 0.4)
for (i in 1:n.subjects) {
  x_i <- sort(runif(n[i], -1, 1))
  y_i <- a[i] * x_i ^ 2 + b[i] * x_i + c[i]
  x[[i]] <- x_i; y[[i]] <- y_i
}
x <- unlist(x)
y <- unlist(y)
## add noise to the true trajectories
y <- y + rnorm(length(y), sd = 0.1)
## compose a data frame in long format
id <- rep.int(1:n.subjects, n)
dat <- data.frame(x = x, y = y, id = as.factor(id))

## fit a penalized splines mixed-effects model
mgcv.form <- y ~ s(x, bs = 'ps', k = 8) +
                 s(x, id, bs = 'fs', xt = 'ps', k = 8, m = c(2, 1))
fit <- psme(mgcv.form, data = dat)

## evaluate population smooth function and subject smooth functions
xp <- seq.int(-1, 1, length.out = 51)
pop.smooth <- EvalSmooth(fit$smooth[[1]], new.x = xp)
sub.smooth <- EvalSmooth(fit$smooth[[2]], new.x = xp)
```

```
smooth <- pop.smooth + sub.smooth + fit$pcoef[["(Intercept)"]]
matplot(xp, smooth, type = "l", lty = 1, xlab = "x", ylab = "fitted trajectories")
points(dat, pch = 20, col = 8)
```

Index