Package ‘BayesSIM’

November 13, 2025

Type Package

Title Integrated Interface of Bayesian Single Index Models using 'nimble'

Version 0.1.5

Description Provides tools for fitting Bayesian single index models with flexible choices of priors for both the index and the link function. The package implements model estimation and posterior inference using efficient MCMC algorithms built on the 'nimble' framework, allowing users to specify, extend, and simulate models in a unified and reproducible manner. The following methods are implemented in the package: Antoniadis et al. (2004) <`https://www.jstor.org/stable/24307224`>, Wang (2009) <doi:10.1016/j.csda.2008.12.010>, Choi et al. (2011) <doi:10.1080/10485251003768019>, Dhar
BA1170>, McGee et al. (2023) <doi:10.1111/biom.13569>.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 4.1), nimble, ggplot2

Imports magrittr, patchwork, tidyr, coda, mvtnorm, methods, MASS

RoxygenNote 7.3.3

NeedsCompilation no

Author Seowoo Jung [aut, cre],
Eun-kyung Lee [aut]

Maintainer Seowoo Jung <`jsw1347@ewha.ac.kr`>

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository CRAN

Date/Publication 2025-11-13 06:50:02 UTC

Contents

`alphaTheta` *One-to-one polar transformation*

Description

The **nimble** function that converts `theta` (angles in radians) of length $p-1$ into a unit-norm vector $\alpha \in \mathbb{R}^p$ on the unit sphere $\mathbb{S}^{p-1}$ using one-to-one polar transformation. For numerical stability, the result is renormalized, and for identification the sign is flipped so that $\alpha_1 \geq 0$.

Usage

```
alphaTheta(theta)
```

Arguments

`theta`	Numeric vector of angles (in radians) of length $p-1$, which parameterize a point on $\mathbb{S}^{p-1}$.

Details

Let $p = \mathrm{length}(\theta) + 1$. The mapping is

$$\begin{aligned}
\alpha_1 &= \sin(\theta_1), \\
\alpha_i &= \Big(\prod_{j=1}^{i-1} \cos(\theta_j)\Big) \sin(\theta_i), \quad i = 2, \dots, p-1, \\
\alpha_p &= \prod_{j=1}^{p-1} \cos(\theta_j).
\end{aligned}$$

The vector is then scaled to unit length, $\alpha \leftarrow \alpha / \|\alpha\|_2$. Finally, if $\alpha_1 < 0$, the vector is negated so that $\alpha_1 \geq 0$, restricting to a single hemisphere to avoid sign indeterminacy.

Typical angle domains (not enforced here) are: $\theta_1, \dots, \theta_{p-1} \in [0, \pi]$.

Value

A numeric vector α of length $p = \mathrm{length}(\theta) + 1$ with unit Euclidean norm and $\alpha_1 \geq 0$.

See Also

`gpPolar`, `gpPolarHigh`, `predict.bsimGp`

`BayesSIM` *Integrated function for Bayesian single-index regression*

Description

Fits a single–index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ in one function.

Usage

```
BayesSIM(
  x,
  y,
  indexprior = "fisher",
  link = "bspline",
  prior = NULL,
  init = NULL,
  sampling = TRUE,
  fitted = TRUE,
  method = "FB",
  lowerB = NULL,
  upperB = NULL,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
```

```
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response numeric vector/matrix. Other types are not available.
`indexprior`	Index vector prior among `"fisher"` (default), `"sphere"`, `"polar"`, `"spike"`.
`link`	Link function among `"bspline"` (default), `"gp"`
`prior`	List of prior hyperparameters of index, link function, and sigma2. For further details, refer to `help()` of designated function.
`init`	List of initial values of index, link function, and sigma2. For further details, refer to `help()` of designated function.
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("Xlin", "linkFunction", "beta")` is monitored for prediction.
`method`	Character, gpSphere model has 3 different types of sampling method, fully Bayesian method (`"FB"`), empirical Bayes approach (`"EB"`), and empirical Gibbs sampler (`"EG"`). Assign one sampler method. Empirical sampling approach is recommended in high-dimensional data. By default, fully Bayesian approach is assigned.
`lowerB`	This parameter is only for gpSphere model. Numeric vector of element-wise lower bounds for the `"L-BFGS-B"` method. When the empirical Bayes or Gibbs sampler method is used, the marginal likelihood is optimized via `optim(method = "L-BFGS-B")`. The vector must be ordered as `c(index vector, lengthscale, amp, sigma2)`; note that `sigma2` is included only for the empirical Bayes method (omit it for Gibbs). By default, the lower bounds are `-1` for the index vector and `-1e2` for logarithm of `lengthscale`, `amp`, and (if present) `sigma2`.
`upperB`	This parameter is only for gpSphere model. Numeric vector of element-wise upper bounds for the `"L-BFGS-B"` method. When the empirical Bayes or Gibbs sampler method is used, the marginal likelihood is optimized via `optim(method = "L-BFGS-B")`. The vector must be ordered as `c(index vector, lengthscale, amp, sigma2)`; note that `sigma2` is included only for the empirical Bayes method (omit it for Gibbs). By default, the upper bounds are `1` for the index vector and `1e2` for logarithm of `lengthscale`, `amp`, and (if present) `sigma2`.
`monitors2`	Optional character vector of additional monitor nodes. Available: `c("Xlin", "k", "knots", "beta")`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).

`thin2`	Integer. Optional thinning for `monitors2` (default 1).
`nchain`	Integer. Number of MCMC chains (default 1). If >1, different initial values are assigned for each chain.
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Integrated function for Bayesian single-index model. Default model is von-Mises Fisher distribution for index vector with B-spline link function.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of samples with `coda` `mcmc` object. ν(spike-and slab prior), θ, σ^2, `monitors2` variables are included.

`fitted` If `fitted` = `TRUE`, in-sample fitted values is given.

`input` List of data,input values for prior and initial values, and computation time without compiling.

See Also

`bsFisher()`, `bsSphere()`, `bsPolar()`, `bsSpike()`, `gpFisher()`, `gpSphere()`, `gpPolar()`, `gpSpike()`

Examples

```
set.seed(123)
n <- 100; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One-call version
fit <- BayesSIM(X, y, indexprior = "sphere", link = "bspline")

# Split version
models <- BayesSIM(X, y, indexprior = "sphere", link = "bspline", sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                    nchains = 1, setSeed = TRUE, init = models$input$init,
                    summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`bsFisher` *Bayesian single-index regression with B-spline link and von Mises-Fisher prior*

Description

Fits a single-index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the link $f(\cdot)$ is represented by B-spline and the index vector θ has von Mises–Fisher prior.

Usage

```
bsFisher(
  x,
  y,
 prior = list(index = list(direction = NULL, dispersion = 150), link = list(basis =
  list(df = 21, degree = 2, delta = 0.001), beta = list(mu = NULL, cov = NULL)), sigma2
   = list(shape = 0.001, rate = 100)),
  init = list(index = NULL, link = list(beta = NULL), sigma2 = 0.01),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response vector/matrix.
`prior`	Optional named list of prior settings with sublists: `index` von Mises–Fisher prior for the projection vector θ. `direction` is a unit direction vector of the von Mises–Fisher distribution. If `direction` is `NULL`, the estimated vector from projection pursuit regression is assigned. `dispersion` is the concentration parameter $c_{\text{prior}} > 0$. (default `150`) `link` B-spline basis and coefficient of B-spline setup. 1. `basis` For the basis of B-spline, `df` is the number of basis functions (default `21`), `degree` is the spline degree (default `2`) and `delta` is a small jitter for boundary-knot spacing control (default `0.001`). 2. `beta` For the coefficient of B-spline, multivariate normal prior is assigned with mean `mu`, and covariance `cov`. By default, $\mathcal{N}_p(0, \mathrm{I}_p)$

	`sigma2` Error-variance prior hyperparameters. An Inverse-Gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 0.001, rate = 100`)
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior. `index` Initial unit index vector θ. By default, the vector is ranomly sampled from the von Mises–Fisher prior. `link` Initial spline coefficients(`beta`). By default, $\left(X_\theta^\top X_\theta + \rho\,\mathrm{I}\right)^{-1} X_\theta^\top Y$ is computed, where X_θ is the B-spline basis design matrix. `sigma2` Initial scalar error variance (default `0.01`).
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("Xlin", "linkFunction", "beta")` is monitored for prediction.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- bsFisher(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single–index model uses a m-order polynomial spline with k interior knots as follows: $f(t) = \sum_{j=1}^{1+m+k} B_j(t)\,\beta_j$ on $[a,b]$ with $t_i = x_i'\theta, i = 1, \cdots, n$ and $||\theta||_2 = 1$. $\{\beta_j\}_{j=1}^{m+k}$ are spline coefficient and a_θ and b_θ are boundary knots where $a = min(t_i, i = 1, \cdots, n) - \delta$, and $b = max(t_i, i = 1, \cdots, n) + \delta$.

Priors

- von Mises–Fisher prior on the index θ: direction `prior$index$direction`, concentration `prior$index$dispersion`.
- Inverse-Gamma prior on σ^2: $\sigma^2 \sim \mathrm{IG}(a_\sigma, b_\sigma)$.
- Conditional on θ and σ^2, the link coefficients follow $\beta = (\beta_1, \ldots, \beta_K)^\top \sim \mathcal{N}\big(\hat{\beta}_\theta,\, \sigma^2 (X_\theta^\top X_\theta)^{-1}\big)$.

Sampling Random walk metropolis algorithm is used for index vector θ. Given θ, σ^2 and β are sampled from posterior distribution. Further sampling method is described in Antoniadis et al.(2004).

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted` = `TRUE`, in-sample fitted values is given.

`input` List of data,input values for prior and initial values, and computation time without compiling.

References

Antoniadis, A., Grégoire, G., & McKeague, I. W. (2004). Bayesian estimation in single-index models. *Statistica Sinica*, 1147-1164.

Hornik, K., & Grün, B. (2014). movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58, 1-31.

Examples

```
set.seed(123)
n <- 200; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One tool version
fit <- bsFisher(X, y)

# Split version
models <- bsFisher(X, y, sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                   nchains = 1, setSeed = TRUE, inits = models$input$init,
                   summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`bsPolar` *Bayesian single-index regression with B-spline link and one-to-one polar transformation*

Description

Fits a single-index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the link $f(\cdot)$ is represented by B-spline link function and the index vector θ is parameterized on the unit sphere via a one-to-one polar transformation.

Usage

```
bsPolar(
  x,
  y,
  prior = list(index = list(psi = list(alpha = NULL)), link = list(basis = list(df = 21,
    degree = 2, delta = 0.001), beta = list(mu = NULL, cov = NULL)), sigma2 = list(shape
    = 0.001, rate = 100)),
  init = list(index = list(psi = NULL), link = list(beta = NULL), sigma2 = 0.01),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response vector/matrix.
`prior`	Optional named list of prior settings with sublists: `index` `psi` is polar angle and rescaled Beta distribution on $[0, \pi]$ is assigned. Only shape parameter `alpha` of $p-1$ dimension vector is needed since rate parameters is computed to satisfy $\theta_{j,\text{MAP}}$. By default, the shape parameter for each element of the polar vector is set to `5000`. `link` B-spline basis and coefficient of B-spline setup. 1. `basis` For the basis of B-spline, `df` is the number of basis functions (default `21`), `degree` is the spline degree (default `2`) and `delta` is a small jitter for boundary-knot spacing control (default `0.001`). 2. `beta` For the coefficient of B-spline, multivariate normal prior is assigned with mean `mu`, and covariance `cov`. By default, $\mathcal{N}_p(0, \mathrm{I}_p)$ `sigma2` Error-variance prior hyperparameters. An Inverse-Gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 0.001, rate = 100`)
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior. `index` Initial vector of polar angle `psi` $(p-1)$. Each element of angle is between 0 and π.

`link` Initial spline coefficients(`beta`). By default, $\left(X_\theta^\top X_\theta + \rho\,\mathrm{I}\right)^{-1} X_\theta^\top Y$ is computed, where X_θ is the B-spline basis design matrix.

`sigma2` Initial scalar error variance (default `0.01`).

`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("Xlin", "linkFunction", "beta")` is monitored for prediction.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- bsPolar(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single–index model uses a m-order polynomial spline with k interior knots as follows: $f(t) = \sum_{j=1}^{1+m+k} B_j(t)\,\beta_j$ on $[a,b]$ with $t_i = x_i'\theta, i = 1,\cdots,n$ and $\|\theta\|_2 = 1$. $\{\beta_j\}_{j=1}^{m+k}$ are spline coefficient and a_θ and b_θ are boundary knots where $a = min(t_i, i = 1,\cdots,n) - \delta$, and $b = max(t_i, i = 1,\cdots,n) + \delta$. θ lies on the unit sphere ($\|\theta\|_2 = 1$) to ensure identifiability and is parameterized via a one-to-one polar transformation with angle $\psi_1, ..., \psi_{p-1}$. Sampling is performed on the angular parameters ψ defining the index vector.

Priors

- Index vector: Uniform prior on the unit sphere ($\|\theta\|_2 = 1$).
- Inverse-Gamma prior on σ^2: $\sigma^2 \sim \mathrm{IG}(a_\sigma, b_\sigma)$.
- Conditional on θ and σ^2, the link coefficients follow $\beta = (\beta_1, \ldots, \beta_K)^\top \sim \mathcal{N}\big(\hat{\beta}_\theta,\, \sigma^2 (X_\theta^\top X_\theta)^{-1}\big)$.

Sampling Samplers are automatically assigned by nimble.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted = TRUE`, in-sample fitted values is given.

`input` List of data,input values for prior and initial values, and computation time without compiling.

References

Antoniadis, A., Grégoire, G., & McKeague, I. W. (2004). Bayesian estimation in single-index models. *Statistica Sinica*, 1147-1164.

Hornik, K., & Grün, B. (2014). movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58, 1-31.

Dhara, K., Lipsitz, S., Pati, D., & Sinha, D. (2019). A new Bayesian single index model with or without covariates missing at random. *Bayesian analysis*, 15(3), 759.

Examples

```
set.seed(123)
n <- 200; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One tool version
fit <- bsPolar(X, y)

# Split version
models <- bsPolar(X, y, sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                    nchains = 1, setSeed = TRUE, inits = models$input$init,
                    summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`bsSphere` *Bayesian single-index regression with B-spline link and half-unit sphere prior*

Description

Fits a single-index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the link $f(\cdot)$ is represented by B-spline link and the index vector θ is on half-unit sphere.

Usage

```
bsSphere(
  x,
  y,
  prior = list(index = list(nu = list(r1 = 1, r2 = 1)), link = list(knots = list(lambda_k
    = 5, maxknots = NULL), basis = list(degree = 2), beta = list(mu = NULL, tau = NULL,
      Sigma0 = NULL)), sigma2 = list(shape = 0.001, rate = 0.001)),
```

```
  init = list(index = list(nu = NULL, index = NULL), link = list(k = NULL, knots = NULL,
    beta = NULL), sigma2 = 0.01),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response vector/matrix.
`prior`	Optional named list of prior settings with sublists: `index` `nu` is binary inclusion indicators prior for variable selection in the index: `list(r1, r2)` gives the Beta hyperprior on the Bernoulli–inclusion probability w, inducing $p(\nu) \propto \mathrm{Beta}(r_1 + n_\nu, r_2 + p - n_\nu)$ (default `r1 = 1, r2 = 2`). `link` B-spline knots, basis and coefficient setup. 1. `knots` Free-knot prior for the spline. `lambda_k` is the Poisson mean for the number of interior knot k (default `5`). `maxknots` is the maximum number of interior knots. If `maxknots` is `NULL`, the number of interior knots is randomly drawn from a Poisson distribution. 2. `basis` For the basis of B-spline, `degree` is the spline degree (default `2`). 3. `beta` For the coefficient of B-spline, conjugate normal prior on β with covariance $\tau\Sigma_0$ is assigned. By default, `mu` is a zero vector, `tau` is set to the sample size, and `Sigma0` is the identity matrix of dimension $1 + k + m$, where k is the number of interior knots and m is the spline order (degree + 1). `sigma2` Error-variance prior hyperparameters. An Inverse-Gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 0.001, rate = 0.001`). Small values for shape and rate parameters yield a weakly-informative prior on σ^2.
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior. `index` `nu` is binary vector indicating active predictors for the index. `index` is initial unit-norm index vector θ (automatically normalized if necessary, with the first nonzero element made positive for identifiability). The vector length must match the number of columns in data `x`. Ideally, positions where `nu` has a value of 1 should correspond to nonzero elements in θ; elements corresponding to `nu` = 0 will be set to zero.

	`link` `k` is initial number of interior knots. `knots` is initial vector of interior knot positions in $[0, 1]$, automatically scaled to the true boundary. Length of this vector should be equal to the initial value of `k`. `beta` is initial vector of spline coefficients. Length should be equal to the initial number of basis functions with intercept $(1 + k + m)$. `sigma2` Initial scalar error variance. (default `0.01`)
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("linkFunction", "beta", "k", "knots", "numBasis", "a_alpha", "b_alpha", "Xlin")` is monitored for prediction.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- bsSphere(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single–index model uses a m-order polynomial spline with k interior knots and intercept as follows: $f(t) = \sum_{j=1}^{1+m+k} B_j(t)\,\beta_j$ on $[a, b]$ with $t_i = x_i'\theta, i = 1, \cdots, n$ and $\|\theta\|_2 = 1$. $\{\beta_j\}_{j=1}^{m+k+1}$ are spline coefficient and a_θ and b_θ are boundary knots where $a = min(t_i, i = 1, \cdots, n)$, and $b = max(t_i, i = 1, \cdots, n)$. Variable selection is encoded by a binary vector ν, equivalently setting components of θ to zero when $\nu_i = 0$.

Priors

- Free-knot prior: $k \sim \text{Poisson}(\lambda_k)$, knot locations $\xi_i, i = 1, ..., k$ via a Dirichlet prior on the scaled interval.
- Beta–Bernoulli hierarchy for ν, yielding a Beta–Binomial prior.
- Spherical prior on the index: uniform on the half-sphere of dimension $n_\nu$with first nonzero positive.
- Conjugate normal–inverse-gamma on (β, σ^2) enables analytic integration and a lower-dimensional marginal target for RJMCMC.

Sampling Posterior exploration follows a hybrid RJMCMC with six move types: add/remove predictor ν, update θ, add/delete/relocate a knot. The θ update is a random-walk Metropolis via local rotations in a two-coordinate subspace; knot moves use simple proposals with tractable acceptance ratios. Further sampling method is described in Wang(2009).

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of ν, θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted = TRUE`, in-sample fitted values is given.

`input` List of data,input values for prior and initial values, and computation time without compiling.

References

Wang, H.-B. (2009). Bayesian estimation and variable selection for single index models. *Computational Statistics & Data Analysis*, 53, 2617–2627.

Hornik, K., & Grün, B. (2014). movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58, 1-31.

Examples

```
set.seed(123)
n <- 200; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One-call version
fit <- bsSphere(X, y)

# Split version
models <- bsSphere(X, y, sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                    nchains = 1, setSeed = TRUE, init = models$input$init,
                    summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`bsSpike` *Bayesian single-index regression with B-spline link and spike-and-slab prior*

Description

Fits a single-index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the link $f(\cdot)$ is represented by B-spline link function and the index vector θ has spike-and-slab prior.

Usage

```
bsSpike(
  x,
  y,
  prior = list(index = list(nu = list(r1 = 1, r2 = 1), index = list(sigma_theta = 0.25)),
   link = list(basis = list(df = 21, degree = 2, delta = 0.01), beta = list(mu = NULL,
    cov = NULL)), sigma2 = list(shape = 0.001, rate = 100)),
  init = list(index = list(pi = 0.5, nu = NULL, index = NULL), link = list(beta = NULL),
    sigma2 = 0.01),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response vector/matrix.
`prior`	Optional named list of prior settings with sublists: `index` Spike and slab prior hyperparameters: Beta-binomial for variable selection indicator ν (default `r1 = 1`, `r2 = 1`), and normal distribution for selected variables θ (default: $N(0, \sigma_\theta^2)$) `link` B-spline basis and coefficient of B-spline setup. 1. `basis` For the basis of B-spline, `df` is the number of basis functions (default `21`), `degree` is the spline degree (default `2`) and `delta` is a small jitter for boundary-knot spacing control (default `0.01`). 2. `beta` For the coefficient of B-spline, multivariate normal prior is assigned with mean `mu`, and covariance `cov`. By default, $\mathcal{N}_p\big(0, \mathrm{I}_p\big)$ `sigma2` Error-variance prior hyperparameters. An Inverse-Gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 0.001`, `rate = 100`)
`init`	Optional named list of initial values: `index` 1. `pi` Initial selecting variable probability. (default: `0.5`) 2. `nu` Initial vector of inclusion indicators . By default, each `nu` is randomly drawn by $Bernoulli(1/2)$

3. `index` Initial vector of index. By default, each element of indedx vector, which is chosen by nu, is proposed by normal distribution.

`link` Initial spline coefficients (`beta`). By default, $\left(X_\theta^\top X_\theta + \rho \mathrm{I}\right)^{-1} X_\theta^\top Y$ is computed, where X_θ is the B-spline basis design matrix.

`sigma2` Initial scalar error variance (default `0.01`).

`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("Xlin", "linkFunction", "beta")` is monitored for prediction.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- bsSpike(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single–index model uses a m-order polynomial spline with k interior knots as follows: $f(t) = \sum_{j=1}^{1+m+k} B_j(t)\,\beta_j$ on $[a,b]$ with $t_i = x_i'\theta, i = 1, \cdots, n$ and $\|\theta\|_2 = 1$. $\{\beta_j\}_{j=1}^{m+k}$ are spline coefficient and a_θ and b_θ are boundary knots where $a = min(t_i, i = 1, \cdots, n) - \delta$, and $b = max(t_i, i = 1, \cdots, n) + \delta$. θ is a p-dimensional index vector subject to a spike-and-slab prior for variable selection with binary indicator variable ν.

Priors

- Slab coefficients θ: Gaussian $N(0, \sigma_\theta^2)$.
- Inclusion indicators ν: Bernoulli(π).
- Inclusion probability π: Beta(r_1, r_2).
- Inverse-Gamma prior on σ^2: $\sigma^2 \sim \mathrm{IG}(a_\sigma, b_\sigma)$.
- Conditional on θ and σ^2, the link coefficients follow $\beta = (\beta_1, \ldots, \beta_K)^\top \sim \mathcal{N}\big(\hat\beta_\theta,\, \sigma^2 (X_\theta^\top X_\theta)^{-1}\big)$.

Sampling Samplers are automatically assigned by nimble.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of ν, θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted = TRUE`, in-sample fitted values is given.

`input` List of data,input values for prior and initial values, and computation time without compiling.

References

Antoniadis, A., Grégoire, G., & McKeague, I. W. (2004). Bayesian estimation in single-index models. *Statistica Sinica*, 1147-1164.

Hornik, K., & Grün, B. (2014). movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58, 1-31.

McGee, G., Wilson, A., Webster, T. F., & Coull, B. A. (2023). Bayesian multiple index models for environmental mixtures. *Biometrics*, 79(1), 462-474.

Examples

```
set.seed(123)
n <- 200; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One tool version
fit <- bsSpike(X, y)

# Split version
models <- bsSpike(X, y, sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 10000, nburnin = 1000, thin = 1,
                    nchains = 1, setSeed = TRUE, inits = models$input$init,
                    summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`compileModelAndMCMC` *Compile a NIMBLE model and its MCMC*

Description

Compiles a NIMBLE model object and a corresponding (uncompiled) MCMC algorithm and returns the compiled pair.

Usage

```
compileModelAndMCMC(fullmodel)
```

Arguments

`fullmodel`	Class "bsimSpline" and "bsimGp" object with sampling = FALSE

Details

The function first compiles `fullmodel$model` with `nimble::compileNimble()`, then compiles `fullmodel$sampler` with `nimble::compileNimble(project = model)`, where `model` is the uncompiled model used to build the sampler. The compiled model and compiled MCMC are returned as a list.

Value

A list with two elements:

`model`	the compiled NIMBLE model (external pointer object).
`mcmc`	the compiled MCMC function/algorithm bound to the model.

See Also

`nimbleModel`, `configureMCMC`, `buildMCMC`, `compileNimble`, `runMCMC`

Examples

```
# Split version
models <- bsplineFisher(DATA1$X, DATA1$y, sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                   nchains = 1, setSeed = TRUE, inits = models$input$initial,
                   summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`concrete` *UCI Concrete Compressive Strength (n = 1030, p = 8)*

Description

Concrete compressive strength dataset from the UCI Machine Learning Repository. No missing variables and there are 8 quantitative inputs and 1 quantitative output.

Usage

```
data(concrete)
```

Format

Numeric data.frame of size 1030 × 8.

cement Numeric. Cement content (kg/m^3).

blast_furnace_slag Numeric. Blast furnace slag (kg/m^3).

fly_ash Numeric. Fly ash (kg/m^3).

water Numeric. Mixing water (kg/m^3).

superplasticizer Numeric. Superplasticizer (kg/m^3).

coarse_aggreate Numeric. Coarse aggregate (kg/m^3).

fine_aggregate Numeric. Fine aggregate (kg/m^3).

age Numeric. Curing age (days; typically 1–365).

strength Numeric. Concrete compressive strength (MPa).

Details

Source Concrete Compressive Strength in UCI Machine Learning Repository. This data is integrated by experimental data from 17 different sources to check the realiability of the strength. This dataset compiles experimental concrete mixes from multiple studies and is used to predict compressive strength and quantify how mixture ingredients and curing age affect that strength.

Variables.

- Cement, Blast Furnace Slag, Fly Ash, Water, Superplasticizer, Coarse Aggregate, Fine Aggregate: quantities in kg per m^3 of mixture.
- Age: curing time in days (1–365).
- Target(strength): compressive strength in MPa.

References

Yeh, I. (1998). Concrete Compressive Strength [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5PK67.

Yeh, I. (1998). Modeling of strength of high-performance concrete using artificial neural networks. Cement and Concrete research, 28(12), 1797-1808.

Examples

```
data(concrete)
str(concrete)
plot(density(concrete$strength), main = "Concrete compressive strength (MPa)")
```

`DATA1` *Simulated single–index data (n = 200, p = 4)*

Description

Synthetic data from a single–index model $y = f(X'\theta) + \varepsilon$ with $f(u) = u^2 \exp(u)$ and $\varepsilon \sim N(0, \sigma^2)$. The index vector is $\theta = (2, 1, 1, 1)/\|(2, 1, 1, 1)\|_2$ and $\sigma = 0.5$.

Usage

```
data(DATA1)
```

Format

X Numeric matrix of size 200 × 4, entries i.i.d. $Unif(-1, 1)$.

y Numeric vector of length 200.

Examples

```
data(DATA1)
str(DATA1)
```

`expcov_gpPolar` *Covariance kernel of OU process*

Description

The **nimble** function that constructs an OU covariance matrix on the $t = X'\theta$. The (i, j) entry is $K_{ij} = \exp\{-\kappa \,|\, \mathrm{t}_i - \mathrm{t}_j \,|\}$, symmetrized explicitly and stabilized with a small diagonal jitter.

Usage

```
expcov_gpPolar(vec, kappa)
```

Arguments

`vec`	Numeric vector of input locations. $t = X'\theta$ is the main input value for the single-index model.
`kappa`	Non-negative numeric scalar range/decay parameter. Larger values imply faster correlation decay.

Details

The OU kernel (a Matérn kernel with smoothness $\nu = 1/2$) induces an exponential correlation that decays with the absolute distance. After filling the matrix, the function enforces symmetry via $(K + K')/2$ and adds 10^{-4} to the diagonal to improve numerical conditioning in downstream linear algebra.

Value

A numeric $n \times n$ covariance matrix, where n is the length of input vector `vec`.

See Also

`gpPolar`, `gpPolarHigh`, `predict.bsimGp`

`expcov_gpSphere` *Covariance kernel with squared-exponential for* `gpSphere()`

Description

A **nimble** function that constructs a covariance matrix on $t = X'\theta$ using a squared–exponential Gaussian kernel with amplitude η and length-scale parameter l. Each entry is defined as $K_{ij} = \eta \exp\{-(t_i - t_j)^2/l\}, \quad i, j = 1, \dots, n$, symmetrized explicitly and stabilized with a small diagonal jitter term.

Usage

```
expcov_gpSphere(vec, l, amp)
```

Arguments

`vec`	Numeric vector of input locations. $t = X'\theta$ is the main input value for the single-index model.
`l`	Positive numeric scalar controlling the length-scale of the kernel. Larger `l` yields slower decay of correlations.
`amp`	Non-negative numeric scalar specifying the amplitude (variance scale) of the kernel.

Details

For the squared–exponential kernel construction, the covariance matrix is symmetrized using $(K + K')/2$ and a small jitter term (10^{-4}) is added to the diagonal to ensure positive-definiteness and numerical stability. The parameters `amp` and `l` jointly control the amplitude (vertical scale) and smoothness (horizontal scale) of the process.

Value

A numeric $n \times n$ covariance matrix with entries $K_{ij} = \eta \exp\{-(t_i - t_j)^2/l\}$ for $i, j = 1, \cdots, n$, symmetrized and stabilized with a diagonal jitter term 10^{-4}.

See Also

`gpSphere`, `predict.bsimGp`

`expcov_gpSpike` *Covariance kernel with squared-exponential and unit nugget for* `gpSpike()`

Description

A **nimble** function that constructs a covariance matrix on the $t = X'\theta$ using a squared–exponential Gaussian kernel scaled by λ^{-1}, with an added unit nugget on the diagonal. Covariance matrix is $I + K_{ij}$ where $K_{ij} = \lambda^{-1}\exp\{-(\mathrm{t}_i - \mathrm{t}_j)^2\}$ for $i, j = 1, \cdots, n$.

Usage

```
expcov_gpSpike(vec, invlambda)
```

Arguments

`vec`	Numeric vector of input locations. $t = X'\theta$ is the main input value for the single-index model.
`invlambda`	Non-negative numeric scalar scaling the kernel amplitude. Larger values increase both diagonal and off-diagonal entries proportionally.

Details

The off-diagonal structure follows the squared–exponential kernel, producing rapidly decaying correlations as the squared distance grows. The matrix is filled in a symmetric manner and then a unit nugget I is added to the diagonal. The parameter `invlambda` controls the overall signal scale of the kernel component. If a different nugget is desired, adjust externally.

Value

A numeric $n \times n$ covariance matrix with entries $K_{ij} = \lambda^{-1}\exp\{-(\mathrm{t}_i - \mathrm{t}_j)^2\}$ for $i \neq j$, and $K_{ii} = \lambda^{-1} + 1$ where $i, j = 1, \cdots, n$.

See Also

`gpSpike`, `predict.bsimGp`

gpFisher — *Bayesian single-index regression with Gaussian process link and von Mises-Fisher prior*

Description

Fits a single–index model$Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the index θ lies on the unit sphere with von Mises-Fisher prior, and the link $f(\cdot)$ is represented with Gaussian process.

Usage

```
gpFisher(
  x,
  y,
  prior = list(index = list(direction = NULL, dispersion = 150), link = list(lengthscale
    = list(shape = 1/8, rate = 1/8), amp = list(a_amp = -1, b_amp = 1)), sigma2 =
    list(shape = 1, rate = 1)),
  init = list(index = NULL, link = list(lengthscale = 0.1, amp = 1), sigma2 = 1),
  sampling = TRUE,
  fitted = FALSE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response numeric vector/matrix. Other types are not available.
`prior`	Optional named list of prior settings with sublists: `index` von Mises–Fisher prior for the projection vector θ. `direction` is a unit direction vector of the von Mises–Fisher distribution. If `direction` is `NULL`, the estimated vector from projection pursuit regression is assigned. `dispersion` is the concentration parameter $c_{\text{prior}} > 0$. (default `150`) `link` 1. `lenghscale`: Prior of length-scale parameter for covariance kernel. Gamma distribution is assigned for l ($\text{G}(\alpha_l, \beta_l)$). `shape` is shape parameter (default `1/8`) and `rate` is rate parameter of `lenghscale` (default `1/8`) 2. `amp`: Prior of amplitude parameter for covariance kernel. Log-normal distribution is assigned for η: $\log(\eta) \sim \text{N}(a_\eta, b_\eta)$ `a_amp` is mean(default `-1`), and `b_amp` is standard deviation(default `1`)

	`sigma2` Error-variance prior hyperparameters. An inverse-gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 1, rate = 1`)
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior. `index` Initial unit index vector θ. By default, the vector is sampled from the von Mises–Fisher prior. `link` `lenghscale` is initial scalar range parameter. (default: `0.1`) `amp` is initial scalar scale parameter. (default: `1`) `sigma2` Initial scalar error variance. (default: `1`)
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), posterior fitted values are included in the output. Also, if `"sampling = FALSE"`, parameters for prediction(`c("linkFunction", "Xlin", "lengthscale", "amp")`) is additionally monitored.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- gpFisher(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for primary monitors (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single-index model uses Gaussian process with zero mean and and covariance kernel $\eta\text{exp}(-\frac{(t_i-t_j)^2}{l})$ as a link function, where $t_i, t_j, j = 1, \ldots, n$ is index value. Index vector should be length 1.

Priors

- von Mises–Fisher prior on the index θ: direction `prior$index$direction`, concentration `prior$index$dispersion`.
- Covariance kernel: $\eta \sim \text{lognormal}(a_\eta, b_\eta)$, $l \sim \text{G}(\alpha_l, \beta_l)$
- Error variance σ^2: $IG(a_\sigma, b_\sigma)$.

Sampling All sampling parameters' samplers are assigned by nimble.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted` = `TRUE`, summary values of in-sample fitted values are included.

`input` List of data,input values for prior and initial values, and computation time without compiling.

References

Antoniadis, A., Grégoire, G., & McKeague, I. W. (2004). Bayesian estimation in single-index models. *Statistica Sinica*, 1147-1164.

Choi, T., Shi, J. Q., & Wang, B. (2011). A Gaussian process regression approach to a single-index model. *Journal of Nonparametric Statistics*, 23(1), 21-36.

Hornik, K., & Grün, B. (2014). movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58, 1-31.

Examples

```
set.seed(20250818)
N <- 60; p <- 2
x1 <- runif(N, -3, 5)
x2 <- runif(N, -3, 5)
beta1 <- 0.45; beta2 <- sqrt(1-beta1^2)
sigma <- sqrt(0.0036)
xlin <- x1*beta1 + x2*beta2
eta <- 0.1*xlin + sin(0.5*xlin)^2
y <- rnorm(N, eta, sigma)
x <- matrix(c(x1, x2), ncol = 2)

# One-call version
fit <- gpFisher(x = x, y = y, nchain = 3, fitted = TRUE)

# Split version
models <- gpFisher(x = x, y = y, nchain = 1, sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                   nchains = 1, setSeed = TRUE, inits = models$input$init,
                    summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`gpPolar` *Bayesian single-index regression with Gaussian process link and one-to-one polar transformation*

Description

Fits a single–index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the index θ is specified and computed via a one-to-one polar transformation, and the link $f(\cdot)$ is represented with a Gaussian process.

Usage

```
gpPolar(
  x,
  y,
 prior = list(index = list(psi = list(alpha = NULL)), link = list(kappa = list(min_kappa
   = 0.5, max_kappa = 4, grid.width = 0.1)), sigma2 = list(shape = 2, rate = 0.01)),
  init = list(index = list(psi = NULL), link = list(kappa = 2), sigma2 = 0.01),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)

gpPolarHigh(
  x,
  y,
 prior = list(index = list(psi = list(alpha = NULL)), link = list(kappa = list(min_kappa
   = 0.5, max_kappa = 4, grid.width = 0.1)), sigma2 = list(shape = 2, rate = 0.01)),
  init = list(index = list(psi = NULL), link = list(kappa = 2), sigma2 = 0.01),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response numeric vector/matrix. Other types are not available.
`prior`	Optional named list of prior settings with sublists: `index` `psi` is polar angle and rescaled Beta distribution on $[0, \pi]$ is assigned. Only shape parameter `alpha` of $p-1$ dimension vector is needed since rate parameters is computed to satisfy $\theta_{j,\text{MAP}}$. By default, the shape parameter for each element of the polar vector is set to `5000`. `link` Prior for the smoothness parameter `kappa` in the Gaussian process kernel: Prior for κ is discrete uniform of equally spaced grid points in $[\kappa_{\min}, \kappa_{\max}]$.

	`min_kappa` is minimum value of kappa (default `0.5`), `max_kappa` is maximum value of kappa (default `4`), and `grid.width` is space (default `0.1`).
	`sigma2` Error-variance prior hyperparameters. An Inverse-Gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 2, rate = 0.01`)
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior.
	`index` Initial vector of polar angle `psi` with $p-1$ dimension. Each element of angle is between 0 and π.
	`link` Initial scalar scale parameter of covariance kernel `kappa`. (default: `2`)
	`sigma2` Initial scalar error variance. (default: `0.01`)
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("linkFunction", "kappa", "Xlin")` is monitored for prediction.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- gpPolar(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single–index model is specified as $Y_i = f(X_i'\theta) + \epsilon_i$, where the index vector θ lies on the unit sphere with ($\|\theta\|_2 = 1$) with non-zero first component to ensure identifiability and is parameterized via a one-to-one polar transformation with angle $\psi_1, ..., \psi_{p-1}$. Sampling is performed on the angular parameters θ defining the index vector. The link function $f(\cdot)$ is modeled by a Gaussian process prior with zero mean and an Ornstein–Uhlenbeck (OU) covariance kernel $\exp(-\kappa|t_i - t_j|), i, j = 1, \ldots, N$, where κ is a bandwidth (smoothness) parameter and t_i, t_j is index value($t_i = X_i'\theta$).

Priors

- Index vector: Uniform prior on the unit sphere ($\|\theta\|_2 = 1$).
- Bandwidth parameter κ: discrete uniform prior over a fixed grid.
- Error variance σ^2: Inverse–Gamma prior.

Sampling For `gpPolar()`, θ is sampled by Metropolis-Hastings and updated with f, κ is chosen by grid search method that maximizes likelihood, σ^2 are sampled from their full conditional distributions using Gibbs sampling. Since κ is sampled by grid search, more than 5 dimension of variables `gpPolarHigh()` is recommended. For `gpPolarHigh()`, all sampling parameters' samplers are assigned by nimble.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted` = `TRUE`, in-sample fitted values is given.

`input` List of data,input values for prior and initial values, and computation time without compiling.

References

Dhara, K., Lipsitz, S., Pati, D., & Sinha, D. (2019). A new Bayesian single index model with or without covariates missing at random. *Bayesian analysis*, 15(3), 759.

Examples

```
library(MASS)
N <- 100    # Sample Size
p <- 3
mu <- c(0,0,0)
rho <- 0
Cx <- rbind(c(1,rho,rho), c(rho,1,rho), c(rho, rho,1))
X <- mvrnorm(n = N, mu=mu, Sigma=Cx, tol=1e-8)
alpha <- c(1,1,1)
alpha <- alpha/sqrt(sum(alpha^2))
z <- matrix(0,N)
z <- X %*% alpha
z <- z[,1]
sigma <- 0.3
f <- exp(z)
y <- f + rnorm(N, 0, sd=sigma) # adding noise
y <- y-mean(y)
f_all <- f
x_all <- X
y_all <- y
data_frame <- cbind(x_all, y, f)
colnames(data_frame) = c('x1', 'x2', 'x3', 'y','f')

# One-call version
fit1 <- gpPolar(X, y)
fit2 <- gpPolarHigh(X, y)

# Split version
models1 <- gpPolar(X, y, sampling = FALSE)
models2 <- gpPolarHigh(X, y, sampling = FALSE)
Ccompile1 <- compileModelAndMCMC(models1)
Ccompile2 <- compileModelAndMCMC(models2)
mcmc.out1 <- runMCMC(Ccompile1$mcmc, niter = 5000, nburnin = 1000, thin = 1,
```

```
                      nchains = 1, setSeed = TRUE, init = models1$input$init,
                      summary = TRUE, samplesAsCodaMCMC = TRUE)
mcmc.out2 <- runMCMC(Ccompile2$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                      nchains = 1, setSeed = TRUE, init = models2$input$init,
                      summary = TRUE, samplesAsCodaMCMC = TRUE)
```

---

gpSphere *Bayesian single-index regression with Gaussian process link and unit sphere prior*

Description

Fits a single–index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where the index θ lies on the unit sphere, and the link $f(\cdot)$ is represented with Gaussian process.

Usage

```
gpSphere(
  x,
  y,
 prior = list(index = NULL, link = list(lengthscale = list(shape = 1/8, rate = 1/8), amp
    = list(a_amp = -1, b_amp = 1)), sigma2 = list(shape = 1, rate = 1)),
 init = list(index = list(index = NULL), link = list(lengthscale = 0.1, amp = 1), sigma2
    = 1),
  sampling = TRUE,
  fitted = TRUE,
  method = "FB",
  lowerB = NULL,
  upperB = NULL,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response numeric vector/matrix. Other types are not available.
`prior`	Optional named list of prior settings with sublists:
	`index` Nothing to assign.

	`link` 1. `lenghscale`: Prior of length-scale parameter for covariance kernel. Gamma distribution is assigned for l: $\mathrm{G}(\alpha_l, \beta_l)$ `shape` is shape parameter (default `1/8`) and `rate` is rate parameter of lengthscale l. (default `1/8`) 2. `amp`: Prior of amplitude parameter for covariance kernel. Log-normal distribution is assigned for η: $\log(\eta) \sim \mathrm{N}(a_\eta, b_\eta)$ `a_amp` is mean (default `-1`), and `b_amp` is standard deviation (default `1`) `sigma2` Error-variance prior hyperparameters. An inverse-gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 1, rate = 1`)
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior. `index` Initial unit index vector. By default, vector is randomly drawn from normal distribution and standardized. `link` `lenghscale` is initial scalar range parameter. (default: `0.1`) `amp` is initial scalar scale parameter. (default: `1`) `sigma2` Initial scalar error variance. (default: `1`)
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `TRUE` (default), posterior fitted values are included in the output. Also, if `"sampling = FALSE"`, parameters for prediction(`c("linkFunction", "Xlin", "lengthscale", "amp")`) is additionally monitored.
`method`	Character, Gp-uniform model has 3 different types of sampling method, fully Bayesian method (`"FB"`), empirical Bayes approach (`"EB"`), and empirical Gibbs sampler (`"EG"`). Assign one sampler method. Empirical sampling approach is recommended in high-dimensional data. By default, fully Bayesian approach is assigned.
`lowerB`	Numeric vector of element-wise lower bounds for the `"L-BFGS-B"` method. When the empirical Bayes or Gibbs sampler method is used, the marginal likelihood is optimized via `optim(method = "L-BFGS-B")`. The vector must be ordered as `c(index vector, lengthscale, amp, sigma2)`; note that `sigma2` is included only for the empirical Bayes method (omit it for Gibbs). By default, the lower bounds are `-1` for the index vector and `-1e2` for logarithm of `lengthscale`, `amp`, and (if present) `sigma2`.
`upperB`	Numeric vector of element-wise upper bounds for the `"L-BFGS-B"` method. When the empirical Bayes or Gibbs sampler method is used, the marginal likelihood is optimized via `optim(method = "L-BFGS-B")`. The vector must be ordered as `c(index vector, lengthscale, amp, sigma2)`; note that `sigma2` is included only for the empirical Bayes method (omit it for Gibbs). By default, the upper bounds are `1` for the index vector and `1e2` for logarithm of `lengthscale`, `amp`, and (if present) `sigma2`.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- gpSphere(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).

`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for primary monitors (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single-index model uses Gaussian process with zero mean and and covariance kernel $\eta\text{exp}(-\frac{(t_i-t_j)^2}{l})$ as a link function, where $t_i, t_j, j = 1, \ldots, n$ is index value. Index vector should be length 1.

Priors

- Index vector: Uniform prior with $||\theta|| = 1$
- Covariance kernel: $\eta \sim$ lognormal(a_η, b_η) , $l \sim$ G(α_l, β_l)
- Error variance σ^2: $IG(a_\sigma, b_\sigma)$

Sampling In the fully Bayesian approach, θ, l, and η are updated via the Metropolis–Hastings algorithm, while f and σ^2 are sampled using Gibbs sampling.

In the empirical Bayes approach, θ, l, η, and σ^2 are estimated by maximum a posteriori (MAP), and f is sampled from its full conditional posterior distribution.

In the empirical Gibbs sampler, θ, l, and η are estimated by MAP, whereas f and σ^2 are sampled via Gibbs sampling.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of θ, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted = TRUE`, summary values of in-sample fitted values are included.

`input` List of input values for prior, initial values and execution time without compiling.

References

Choi, T., Shi, J. Q., & Wang, B. (2011). A Gaussian process regression approach to a single-index model. *Journal of Nonparametric Statistics*, 23(1), 21-36.

Examples

```
set.seed(123)
n <- 200; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One-call version
fit <- gpSphere(X, y, method = "EB")

# Split version
model <- gpSphere(X, y, method = "EB", sampling = FALSE)
Ccompile <- compileModelAndMCMC(model)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                   nchains = 1, setSeed = TRUE, inits = model$input$init,
                    summary = TRUE, samplesAsCodaMCMC = TRUE)
```

---

gpSpike *Bayesian single-index regression with Gaussian process link and spike-and-slab prior*

Description

Fits a single-index model $Y_i \sim \mathcal{N}(f(X_i'\theta), \sigma^2), i = 1, \cdots, n$ where index vector θ has a spike and slab prior and the link $f(\cdot)$ is represented by Gaussian process and the

Usage

```
gpSpike(
  x,
  y,
  prior = list(index = list(r1 = 1, r2 = 1, sigma_theta = 0.25), link =
   list(inv_lambda_shape = 1, inv_lambda_rate = 0.1), sigma2 = list(shape = 0.001, rate
    = 0.001)),
 init = list(index = list(pi = 0.5, nu = NULL, index = NULL), link = list(inv_lambda =
    NULL), sigma2 = NULL),
  sampling = TRUE,
  fitted = TRUE,
  monitors2 = NULL,
  niter = 10000,
  nburnin = 1000,
  thin = 1,
  thin2 = NULL,
```

```
  nchain = 1,
  setSeed = FALSE
)
```

Arguments

`x`	Numeric data.frame/matrix of predictors. Each row is an observation.
`y`	Numeric response numeric vector/matrix. Other types are not available
`prior`	Optional named list of prior settings with sublists: `index` Spike and slab prior hyperparameters: Beta-binomial for variable selection (default `r1 = 1`, `r2 = 1`), and normal distribution for selected variables (default: $N(0, \sigma_\theta^2)$) `link` Gaussian process prior hyperparameters `lambda`: Inverse-Gamma prior is assigned for λ^{-1} (default `inv_lambda_shape = 1`, `inv_lambda_rate = 0.1`) `sigma2` Error-variance prior hyperparameters. An Inverse-Gamma prior is assigned to σ^2 where `shape` is shape parameter and `rate` is rate parameter of inverse gamma distribution. (default `shape = 0.001`, `rate = 100`)
`init`	Optional named list of initial values. If the values are not assigned, they are randomly sampled from prior. `index` 1. `pi`: Initial selecting variable probability. (default: `0.5`) 2. `nu`: Initial vector of inclusion indicators . By default, each `nu` is randomly drawn by $Bernoulli(1/2)$ 3. `index`: Initial vector of index. By default, each element of index vector, which is chosen by nu, is proposed by normal distribution. `link` Initial scalar of lambda (`inv_lambda`) for covariance of Gaussian process. `sigma2` Initial scalar error variance. (default: `0.01`)
`sampling`	Logical. If `TRUE` (default), run MCMC; otherwise return prepared nimble model objects without sampling.
`fitted`	Logical. If `fitted = FALSE`, fitted values are not drawn and only `c("nu", "indexstar", "sigma2")` are monitored. If `fitted = TRUE` (default), fitted values drawn from posterior distribution are included in the output and `c("Xlin", "invlambda")` is additionally monitored for prediction.
`monitors2`	Optional character vector of additional monitor nodes. To check the names of the nodes, set `fit <- gpSpike(x, y, sampling = FALSE)` and then inspect the variable names stored in the model object using `fit$model$getVarNames()`.
`niter`	Integer. Total MCMC iterations (default `10000`).
`nburnin`	Integer. Burn-in iterations (default `1000`).
`thin`	Integer. Thinning for monitors1 (default `1`).
`thin2`	Integer. Optional thinning for `monitors2` (default `1`).
`nchain`	Integer. Number of MCMC chains (default `1`).
`setSeed`	Logical or numeric argument. Further details are provided in runMCMC.

Details

Model The single–index model is specified as $Y_i = f(X_i'\theta) + \epsilon_i$, where θ is a p-dimensional index vector subject to a spike-and-slab prior for variable selection. The link function $f(\cdot)$ is modeled using a Gaussian process prior with zero mean and squared exponential covariance kernel $K(x_1, x_2) = \exp\{-\rho(x_1 - x_2)^T\theta^2\}$, where ρ determines the smoothness of f. The covariance kernel is re-parameterized to $\exp\{-(x_1 - x_2)^T{\theta^*}^2\}$ where $\rho = ||\theta^*||$ and $\theta = ||\theta||^{-1}\theta^*$. Therefore, θ^* is sampled in MCMC.

Priors

- Inclusion indicators ν_l: Bernoulli(π).
- Inclusion probability π: Beta(r_1, r_2).
- Slab coefficients θ_l^*: Gaussian $N(0, \sigma_\theta^2)$.
- GP precision λ^{-1}: Gamma(a_λ, b_λ).
- Error precision $(\sigma^2)^{-1}$: Gamma(a_σ, b_σ).

Sampling A random walk Metropolis algorithm is used to sample λ^{-1} and a Metropolis-Hastings algorithm is used for the main parameters (θ^*, ν). The variance σ^2 is directly sampled from posterior distribution. f is not directly sampled by MCMC.

Value

A `list` typically containing:

`model` Nimble model

`sampler` Nimble sampler

`sampling` Posterior draws of ν, θ^*, σ^2, and nodes for fitted values by default. Variables specified in `monitors2` will be added if provided.

`fitted` If `fitted = TRUE`, in-sample fitted values.

`input` List of input values for prior, initial values and execution time without compiling.

References

McGee, G., Wilson, A., Webster, T. F., & Coull, B. A. (2023). Bayesian multiple index models for environmental mixtures. *Biometrics*, 79(1), 462-474.

Examples

```
set.seed(123)
n <- 200; d <- 4
theta <- c(2, 1, 1, 1); theta <- theta / sqrt(sum(theta^2))
f <- function(u) u^2 * exp(u)
sigma <- 0.5
X <- matrix(runif(n * d, -1, 1), nrow = n)
index_vals <- as.vector(X %*% theta)
y <- f(index_vals) + rnorm(n, 0, sigma)

# One tool version
```

```
fit <- gpSpike(X, as.vector(y))

# Split version
models <- gpSpike(X, as.vector(y), sampling = FALSE)
Ccompile <- compileModelAndMCMC(models)
mcmc.out <- runMCMC(Ccompile$mcmc, niter = 5000, nburnin = 1000, thin = 1,
                   nchains = 1, setSeed = TRUE, init = models$input$init,
                   summary = TRUE, samplesAsCodaMCMC = TRUE)
```

`plot.bsimGp` *Traceplot for bsimGp Results*

Description

Provides traceplot for objects of class `bsimGp`, corresponding to a single-index model with a Gaussian process link function. Z

Usage

```
## S3 method for class 'bsimGp'
plot(x, ...)
```

Arguments

`x`	An object of class `bsimGp`.
`...`	Further arguments passed to `plot`.

Value

Traceplots for MCMC samples are displayed. By default, only the index vector and error variance are included in the summary. If a spike-and-slab prior model, indexstar is parameterized to original index vector.

`plot.bsimSpline` *Traceplot for bsimSpline Results*

Description

Provides traceplot for objects of class `bsimSpline`, corresponding to a single-index model with a Gaussian process link function.

Usage

```
## S3 method for class 'bsimSpline'
plot(x, ...)
```

Arguments

`x`	An object of class `bsimSpline`.
`...`	Further arguments passed to `plot`.

Value

Traceplots for MCMC samples are displayed. By default, only the index vector and error variance are included in the summary.

`predict.bsimGp` *Predict method for* `bsimGp` *objects*

Description

Computes posterior predictive summaries from a fitted single–index GP model (class `bsimGp`). Optionally returns a diagnostic plot and RMSE when observed responses are provided.

Usage

```
## S3 method for class 'bsimGp'
predict(
  object,
  newdata = NULL,
  method = "mean",
  se.fit = FALSE,
  level = 0.95,
  ...
)
```

Arguments

`object`	An object of class `bsimGp` created with MCMC sampling enabled.
`newdata`	A `data.frame` of predictors for prediction. The columns must match the columns of train data. If a column named `"y"` is present, it is treated as the observed response and is used to compute RMSE and to draw fitted plots. If `NULL`, fitted values with train data are summarized.
`method`	A character scalar selecting the point estimator returned in the summary: `"mean"` (posterior mean) or `"median"` (posterior median).
`se.fit`	Logical; if `TRUE`, include posterior standard deviation and interval bounds in the returned summary and shaded area for the interval in the fitted curve plot.
`level`	Numeric scalar in $[0, 1]$; nominal coverage for intervals. By default, `level = 0.95`.
`...`	Further arguments passed to `predict`.

Details

The function first gathers posterior draws and computes predictive draws for each row of `newdata` (or for the training data when `newdata = NULL`). From these draws it forms.:

- posterior mean and median,
- empirical standard deviation (`sd`),
- lower/upper quantiles for a `level` interval (`LB`, `UB`).

If `newdata` supplies a response column `y`, the function also computes the mean squared prediction error (reported as `rmse`) and creates two ggplot objects: (i) True vs Predicted with a 45° line and (ii) Index value vs Response showing the fitted curve. If `se.fit = TRUE`, a shaded area visualizes the `level` interval.

The index used in the fitted-curve plot is formed by projecting predictors onto the estimated index vector extracted from `summary(object)`.

Value

A `list` with components:

- `summary`: a `data.frame` with columns matching `method`, either `mean` or `median`; if `se.fit = TRUE` the columns `sd`, `LB`, `UB` are included.
- `plot`: a combined `ggplot` object with the fitted plots when an observed `y` is available, otherwise `NULL`.
- `rmse`: mean squared prediction error when `y` is available, otherwise `NULL`.

See Also

`summary.bsimGp`

`predict.bsimSpline` *Predict method for* `bsimSpline` *objects*

Description

Computes posterior predictive summaries from a fitted single–index B-spline model (class `bsimSpline`). Optionally returns a diagnostic plot and RMSE when observed responses are provided.

Usage

```
## S3 method for class 'bsimSpline'
predict(
  object,
  newdata = NULL,
  method = "mean",
  se.fit = FALSE,
  level = 0.95,
  ...
)
```

Arguments

`object`	An object of class `bsimSpline` created with MCMC sampling enabled.
`newdata`	A `data.frame` of predictors for prediction. The columns must match the columns of train data. If a column named `"y"` is present, it is treated as the observed response and is used to compute RMSE and to draw fitted plots. If `NULL`, fitted values with train data are summarized.
`method`	A character scalar selecting the point estimator returned in the summary: `"mean"` (posterior mean) or `"median"` (posterior median).
`se.fit`	Logical; if `TRUE`, include posterior standard deviation and interval bounds in the returned summary and shaded area for the interval in the fitted curve plot.
`level`	Numeric scalar in $[0, 1]$; nominal coverage for intervals. By default, `level = 0.95`.
`...`	Further arguments passed to `predict`.

Details

The function first gathers posterior draws and computes predictive draws for each row of `newdata` (or for the training data when `newdata = NULL`). From these draws it forms.:

- posterior mean and median,
- empirical standard deviation (`sd`),
- lower/upper quantiles for a `level` interval (`LB`, `UB`).

If `newdata` supplies a response column `y`, the function also computes the mean squared prediction error (reported as `rmse`) and creates two ggplot objects: (i) True vs Predicted with a 45° line and (ii) Index value vs Response showing the fitted curve. If `se.fit = TRUE`, a shaded area visualizes the `level` interval.

The index used in the fitted-curve plot is formed by projecting predictors onto the estimated index vector extracted from `summary(object)`.

Value

A `list` with components:

- `summary`: a `data.frame` with columns matching `method`, either `mean` or `median`; if `se.fit = TRUE` the columns `sd`, `LB`, `UB` are included.
- `plot`: a combined `ggplot` object with the fitted plots when an observed `y` is available, otherwise `NULL`.
- `rmse`: mean squared prediction error when `y` is available, otherwise `NULL`.

See Also

`summary.bsimSpline`

`SplineState` *Additional functions for spline*

Description

Additional functions for spline

Usage

```
SplineState
```

Format

An object of class `list` of length 1.

`summary.bsimGp` *Summarize bsimGp Results*

Description

Provides a `summary` method for objects of class `bsimGp`, corresponding to a single-index model with a Gaussian process link function.

Usage

```
## S3 method for class 'bsimGp'
summary(object, verbose = TRUE, ...)
```

Arguments

`object`	An object of class `bsimGp`.
`verbose`	Logical. If `TRUE`, the summary values for all chain is printed.
`...`	Further arguments passed to `summary`.

Details

A `list` of summary statistics for MCMC samples. Each row corresponds to a model parameter, and columns report the statistics.

Value

The function summarizes posterior MCMC samples by reporting key statistics, including:

- Posterior mean and median
- Empirical standard deviation
- 95% credible interval (lower and upper quantiles)
- Potential scale reduction factor (`gelman`) for multiple chains
- Effective sample size (`ESS`)

By default, only the index vector and error variance are included in the summary. If a spike-and-slab prior is used, the indicator vector (`nu`) is also summarized. Note that the potential scale reduction factor for `nu` can be reported as `NaN` or `Inf`, since the indicator rarely changes during the MCMC run.

See Also

`gelman.diag`, `effectiveSize`

`summary.bsimSpline` *Summarize bsimSpline Results*

Description

Provides a `summary` method for objects of class `bsimSpline`, corresponding to a single-index model with a Gaussian process link function.

Usage

```
## S3 method for class 'bsimSpline'
summary(object, verbose = TRUE, ...)
```

Arguments

`object`	An object of class `bsimSpline`.
`verbose`	Logical. If `TRUE`, the summary values for all chain is printed.
`...`	Further arguments passed to `summary`.

Details

A `data.frame` of summary statistics for MCMC samples. Each row corresponds to a model parameter, and columns report the statistics.

Value

The function summarizes posterior MCMC samples by reporting key statistics, including:

- Posterior mean and median
- Empirical standard deviation
- 95% credible interval (lower and upper quantiles)
- Potential scale reduction factor (`gelman`) for multiple chains
- Effective sample size (`ESS`)

By default, only the index vector and error variance are included in the summary. If a uniform sphere prior is used, the indicator vector (`nu`) is also summarized. Note that the potential scale reduction factor for `nu` can be reported as `NaN` or `Inf`, since the indicator rarely changes during the MCMC run.

See Also

`gelman.diag`, `effectiveSize`

`transX_fisher` *Design matrix of* `bsFisher()`, `bsPolar()`, `bsSpike()`

Description

The **nimble** function that makes $X'\theta$ to B-spline basis design matrix.

Usage

```
transX_fisher(Xlin, df, degree, delta)
```

Arguments

`Xlin`	A numeric vector representing the linear predictor values.
`df`	An integer scalar specifying the total number of basis functions (degrees of freedom).
`degree`	An integer scalar giving the polynomial degree of the B-spline.
`delta`	A numeric scalar that extends the lower and upper boundary knots by `delta` to reduce boundary effects.

Details

The function determines internal knots based on quantiles of `Xlin` and extends the boundary knots by a small `delta`. The resulting basis matrix can be directly used as an input design matrix for spline-based link functions.

This function is intended for internal development purposes and is not designed for direct use by end users.

Value

A numeric matrix containing the B-spline basis expansion of `Xlin` with `df` columns.

See Also

`bsFisher`, `bsPolar`, `bsSpike`, `predict.bsimSpline`

`transX_sp` *Design matrix of* `bsSphere`

Description

A **nimble** function that maps $X'\theta$ to a B-spline basis design matrix with pre-specified candidate knots and boundary knots.

Usage

```
transX_sp(Xlin, degree, knots, k, maxBasis, a_alpha, b_alpha)
```

Arguments

`Xlin`	A numeric vector representing the linear predictor values ($X'\theta$).
`degree`	An integer scalar giving the polynomial degree of the B-spline.
`knots`	A numeric vector of candidate internal knots.
`k`	An integer scalar indicating how many elements of `knots` are active.
`maxBasis`	An integer scalar giving the target number of columns of the returned design matrix. If the raw B-spline basis has fewer than `maxBasis` columns, the result is right-padded with zeros.
`a_alpha`	A numeric scalar specifying the lower boundary knot.
`b_alpha`	A numeric scalar specifying the upper boundary knot.

Details

The function takes a candidate knot vector `knots` and boundary knots set to `c(a_alpha, b_alpha)`. The resulting basis matrix is right-padded with zeros to have exactly `maxBasis` columns, which is convenient for models whose basis dimension may change during estimation but must conform to a fixed maximum width.

This function is intended for internal development purposes and is not designed for direct use by end users.

Value

A numeric matrix of dimension `length(Xlin)` × `maxBasis` containing the B-spline basis (with intercept) constructed from the `k` knots and boundary knots `c(a_alpha, b_alpha)`, zero-padded on the right if needed.

See Also

`bsSphere`, `predict.bsimSpline`

`unitSphere` *Uniform distribution on the unit sphere*

Description

`dunitSphere()` evaluates the density of the uniform distribution on the unit sphere $S^{d-1} = \{x \in \mathbb{R}^d : \|x\| = 1\}$. `runitSphere()` generates one random unit vector in d dimensions using a normalized Gaussian vector.

Usage

```
dunitSphere(x, dim, log = 0)

runitSphere(n, dim)
```

Arguments

`x`	numeric vector of length `dim`. Point in $\mathbb{R}^d$.
`dim`	integer scalar. Dimension of the ambient space. Must be positive.
`log`	logical; If TRUE, probabilities p are given as log(p)
`n`	Integer

Details

- The density of the uniform distribution on the unit sphere in d dimensions is constant and equal to
$$f(x) = 1/\text{SurfaceArea}(S^{d-1})$$
for $x \in S^{d-1}$, where
$$\text{SurfaceArea}(S^{d-1}) = \frac{2\pi^{d/2}}{\Gamma(d/2)}.$$

Value

- `dunitSphere()` : Scalar numeric, the density (or log-density).
- `runitSphere()` : Numeric vector of length `dim` with unit norm. To register distribution for nimble, only one vector is generated.

`vMF` *von-mises Fisher distribution*

Description

Density, and random generation for von-mises Fisher distribution.

Usage

```
dvMFnim(x, theta, log = 0)

rvMFnim(n, theta)
```

Arguments

`x`	vector of direction
`theta`	vector of direction
`log`	logical; If TRUE, probabilities p are given as log(p)
`n`	number of observations

Details

- `dvMFnim(x, theta, log)` evaluates the log-density

  $$\log f(x|\theta) = \theta^\top x - \left[\log I_\nu(\kappa) + \log\Gamma(\nu) - (\nu - 1)\log(\kappa/2)\right],$$

  with $\nu = p/2$, $\kappa = \|\theta\|_2$.
- `rvMFnim(n, theta)` returns one vector on the unit sphere. If $\kappa = 0$, it generates a uniform random direction.

Value

- `dvMFnim()` : scalar numeric (density or log-density).
- `rvMFnim()` : numeric vector of length p. To register distribution for nimble, only one vector is generated.

References

Wood, Andrew TA. "Simulation of the von Mises Fisher distribution." *Communications in statistics-simulation and computation* 23.1 (1994): 157-164.

Hornik, K., & Grün, B. (2014). movMF: an R package for fitting mixtures of von Mises-Fisher distributions. *Journal of Statistical Software*, 58, 1-31.

`Xlinear` *Compute the linear index $X'\theta$*

Description

A **nimble** function that calculates the $X'\theta$ for each observation, given matrix `dataX` and index vector `betavec`. This serves as a basic building block for single-index or regression-type models implemented in **nimble**.

Usage

```
Xlinear(betavec, dataX)
```

Arguments

`betavec`	A numeric vector of regression coefficients of length p.
`dataX`	A numeric matrix of predictors with n rows (observations) and p columns (features).

Details

For a dataset with n observations and p predictors, the function computes linear projection, $X'\theta$. The implementation uses explicit loops to ensure full compatibility with **nimble**'s compiled execution framework.

Value

A numeric vector of length n, representing $X'\theta$.

See Also

`gpPolar`, `gpPolarHigh`, `predict.bsimGp`

Index