Package 'khaos'

April 23, 2026

Type Package

Title Bayesian Sparse Polynomial Chaos Expansion

Version 2.1.1

Description Implements Bayesian polynomial chaos expansion methods for surrogate modeling, uncertainty quantification, and sensitivity analysis. Includes sparse regression-based approaches and adaptive Bayesian models based on reversible-jump Markov chain Monte Carlo. Optional screening and basis-enrichment strategies are provided to improve scalability in moderate to high dimensions.

Imports glmnet, mvtnorm

Suggests knitr, rmarkdown, lhs, testthat (>= 2.1.0)

License BSD_3_clause + file LICENSE

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.2

Author Kellin Rumsey [aut, cre] (ORCID: <https://orcid.org/0000-0002-2989-965X>), J. Derek Tucker [ctb] (ORCID: <https://orcid.org/0000-0001-8844-2169>)

Maintainer Kellin Rumsey <knrumsey@lanl.gov>

Repository CRAN

Date/Publication 2026-04-23 20:30:03 UTC

Contents

`adaptive_khaos` *Bayesian Adaptive Polynomial Chaos Expansion*

Description

A wrapper for adaptive Bayesian polynomial chaos expansion models with different coefficient prior formulations.

Usage

```
adaptive_khaos(
  X,
  y,
  prior_type = "ridge",
  nmcmc = 10000,
  nburn = 9000,
  thin = 1,
  legacy = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`X`	A data frame or matrix of predictors scaled to lie between 0 and 1.
`y`	A numeric response vector of length `nrow(X)`.
`prior_type`	Character string specifying the coefficient-prior formulation. Currently one of `"ridge"` or `"gprior"`.
`nmcmc`	Number of MCMC iterations.
`nburn`	Number of initial MCMC iterations to discard.
`thin`	Keep every `thin`-th retained MCMC sample.
`legacy`	Logical. If `TRUE`, reproduces the historical storage behavior of earlier implementations, which may retain stale values in unused portions of the internal `vars` and `degs` arrays. If `FALSE`, these unused entries are cleared more carefully.
`verbose`	Logical; if `TRUE`, progress messages are printed.
`...`	Additional arguments passed to the prior-specific fitting routine.

Details

Depending on `prior_type`, this function calls either `adaptive_khaos_ridge()` or `adaptive_khaos_gprior()`.

This function dispatches to one of the prior-specific adaptive KHAOS fitting routines:

`prior_type = "ridge"` Calls `adaptive_khaos_ridge`.

`prior_type = "gprior"` Calls `adaptive_khaos_gprior`.

The wrapper exposes only the most commonly used arguments. Additional prior-specific tuning parameters can be supplied through `...`; see `adaptive_khaos_ridge` and `adaptive_khaos_gprior` for full documentation.

Value

An object of class `"adaptive_khaos"` containing the fitted model. The object is a list including sampled basis functions, coefficient samples, variance parameters, and additional MCMC output. The specific structure depends on the chosen `prior_type`.

References

Francom, Devin, and Bruno Sanso. "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." *Journal of Statistical Software* 94.LA-UR-20-23587 (2020).

Rumsey, K.N., Francom, D., Gibson, G., Tucker, J. D., and Huerta, G. (2026). Bayesian Adaptive Polynomial Chaos Expansions. *Stat*, 15(1), e70151.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391 * ((x[1] - 0.4) * (x[2] - 0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)

fit1 <- adaptive_khaos(X, y)
fit2 <- adaptive_khaos(X, y, prior_type = "gprior")
```

---

adaptive_khaos_gprior *Bayesian Adaptive Polynomial Chaos Expansion (Modified g-Prior)*

Description

Adaptive Bayesian polynomial chaos expansion using a modified g-prior. This implementation corresponds to the adaptive Bayesian PCE methodology described in Rumsey et al. (2026)

Usage

```
adaptive_khaos_gprior(
  X,
  y,
  degree = 15,
  order = 5,
  nmcmc = 10000,
  nburn = 9000,
  thin = 1,
  max_basis = 1000,
  a_g = 0.001,
  b_g = 1000,
  zeta = 1,
  g2_sample = "mh",
  g2_init = NULL,
  s2_lower = 0,
  a_sigma = 0,
  b_sigma = 0,
  a_M = 4,
  b_M = 4/length(y),
  move_probs = rep(1/3, 3),
  coin_pars = list(function(j) 1/j, 1, 2, 3),
  degree_penalty = 0,
  legacy = TRUE,
  verbose = TRUE
)

adaptive_khaos2(...)
```

Arguments

X	A data frame or matrix of predictors scaled to be between 0 and 1
y	a response vector
degree	Maximum polynomial degree for each basis function.
order	Maximum order of interaction for each basis function.
nmcmc	Number of MCMC iterations.
nburn	Number of initial MCMC iterations to discard.
thin	Keep every thin samples.
max_basis	Maximum number of basis functions.
a_g, b_g	Prior parameters for global g-prior term (on precision scale)
zeta	Hyperparameter for modified g-prior. Setting zeta = 0 reduces to the usual g-prior.
g2_sample	Character string specifying the method used to sample or update the prior scaling parameter g0^2.

g2_init	Initial calue of g2 (global precision for g-prior). Becomes the only value when g2_sample = "fixed".
s2_lower	Lower bound on process variance (numerically useful for deterministic functions).
a_sigma, b_sigma	Shape/rate parameters for the IG prior on process variance (default is Jefffrey's prior)
a_M, b_M	Shape/rate parameters for the Gamma prior on expected number of basis functions.
move_probs	A 3-vector with probabilities for (i) birth, (ii) death, and (iii) mutation.
coin_pars	A list of control parameters for coinflip proposal
degree_penalty	Increasing this value encourages lower order polynomial terms (0 is no penalization).
legacy	Logical. If TRUE, mimics original implementation behavior (from the emulator comparison paper), which may retain invalid basis function structures across iterations. Defaults to FALSE.
verbose	Logical. Should progress information be printed?
...	additional arguments passed to adaptive_khaos

Details

Implements the RJMCMC algorithm described by Francom & Sanso (2020) for BMARS, modifying it for polynomial chaos basis functions. As an alternative the NKD procedure of Nott et al. (2005), we use a coinflipping procedure to identify useful variables. See writeup (coming soon) for details. The coin_pars argument is an ordered list with elements:

1. a function giving proposal probability for q_0, the expected order of interaction,
2. epsilon, the base weight for each variable (epsilon=Inf corresponds to uniform sampling),
3. alpha, the exponent-learning rate,
4. num_passes a numerical parameter only.

Sampling method for g0^2 depends on g2_sample

"f" Fixed: Do not update g0^2; carry forward the previous value (g2[i] <- g2[i-1]).

"lf" Laplace-Full: Use a Laplace approximation to sample g0^2 under the full marginal likelihood (no orthogonality assumption).

"lo" Laplace-Orthogonal: Use a Laplace approximation to sample g0^2 assuming the orthogonality approximation applies.

"mh" Metropolis-Hastings with full Laplace approximation as proposal. Target is the full posterior density of g0^2.

"mho" Metropolis-Hastings with orthogonal Laplace approximation as proposal. Target is the full posterior density of g0^2. under the orthogonality assumption.

"mhoo" Metropolis-Hastings with orthogonal Laplace approximation as proposal. Target is the posterior density of g0^2 under the orthogonality assumption.

Other proposal notes: Expected order q0 is chosen from 1:order with weights `coin_pars[[1]](1:order)` Degree is chosen from q0:degree with weights `(1:(degree-q0+1))^(-degree_penalty)` and a random partition is created (see helper function).

Two types of change steps are possible (and equally likely) A re-ordering of the degrees A single variable is swapped out with another one (only used when `ncol(X) > 3`)

Value

An object of class `"adaptive_khaos"`. The object is a list containing:

B Matrix of basis functions evaluated at the training inputs.

vars, degs Arrays encoding the variables and polynomial degrees for each basis function across MCMC iterations.

nint, dtot Interaction order and total degree for each basis function.

nbasis Number of basis functions per MCMC iteration.

beta Posterior samples of regression coefficients.

s2 Posterior samples of the noise variance.

lam Posterior samples of the Poisson rate parameter controlling model size.

g2 Posterior samples of the global g-prior scaling parameter.

sum_sq Residual sum of squares at each iteration.

eta Variable inclusion counts used in the proposal distribution.

count_accept, count_propose Acceptance and proposal counts for RJMCMC moves.

prior_type Character string indicating the g-prior formulation.

X, y Training data.

References

Francom, Devin, and Bruno Sanso. "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." Journal of Statistical Software 94.LA-UR-20-23587 (2020).

Nott, David J., Anthony YC Kuk, and Hiep Duc. "Efficient sampling schemes for Bayesian MARS models with many predictors." Statistics and Computing 15 (2005): 93-101.

Rumsey, K.N., Francom, D., Gibson, G., Derek Tucker, J. and Huerta, G., 2026. Bayesian Adaptive Polynomial Chaos Expansions. Stat, 15(1), p.e70151.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- adaptive_khaos(X, y)
```

`adaptive_khaos_ridge` *Bayesian Adaptive Polynomial Chaos Expansion (Ridge Prior)*

Description

Adaptive Bayesian polynomial chaos expansion using a fixed Gaussian (ridge) prior on coefficients. This implementation corresponds to a simplified version of the adaptive Bayesian PCE methodology described in Rumsey et al. (2026)

Usage

```
adaptive_khaos_ridge(
  X,
  y,
  degree = 15,
  order = 5,
  nmcmc = 10000,
  nburn = 9000,
  thin = 1,
  max_basis = 1000,
  tau2 = 10^5,
  s2_lower = 0,
  g1 = 0,
  g2 = 0,
  h1 = 4,
  h2 = 40/length(y),
  move_probs = rep(1/3, 3),
  coin_pars = list(function(j) 1/j, 1, 2, 3),
  degree_penalty = 0,
  legacy = TRUE,
  verbose = TRUE
)
```

Arguments

`X`	A data frame or matrix of predictors scaled to be between 0 and 1
`y`	a response vector
`degree`	Maximum polynomial degree for each basis function.
`order`	Maximum order of interaction for each basis function.
`nmcmc`	Number of MCMC iterations.
`nburn`	Number of initial MCMC iterations to discard.
`thin`	Keep every `thin` samples.
`max_basis`	Maximum number of basis functions.
`tau2`	Prior variance for coefficients

s2_lower	Lower bound on process variance (numerically useful for deterministic functions).
g1, g2	Shape/rate parameters for the IG prior on process variance (default is Jefffrey's prior)
h1, h2	Shape/rate parameters for the Gamma prior on expected number of basis functions.
move_probs	A 3-vector with probabilities for (i) birth, (ii) death, and (iii) mutation.
coin_pars	A list of control parameters for coinflip proposal
degree_penalty	Increasing this value encourages lower order polynomial terms (0 is no penalization).
legacy	Logical. If TRUE, mimics original implementation behavior (from the emulator comparison paper), which may retain invalid basis function structures across iterations. Defaults to FALSE.
verbose	Logical. Should progress information be printed?

Details

Implements the RJMCMC algorithm described by Francom & Sanso (2020) for BMARS, modifying it for polynomial chaos basis functions. As an alternative the NKD procedure of Nott et al. (2005), we use a coinflipping procedure to identify useful variables. See writeup (coming soon) for details. The coin_pars argument is an ordered list with elements:

1. a function giving proposal probability for q_0, the expected order of interaction,
2. epsilon, the base weight for each variable (epsilon=Inf corresponds to uniform sampling),
3. alpha, the exponent-learning rate,
4. num_passes a numerical parameter only.

Other proposal notes: Expected order q0 is chosen from 1:order with weights `coin_pars[[1]](1:order)` Degree is chosen from q0:degree with weights `(1:(degree-q0+1))^(-degree_penalty)` and a random partition is created (see helper function).

Two types of change steps are possible (and equally likely) A re-ordering of the degrees A single variable is swapped out with another one

Value

An object of class `"adaptive_khaos"`. The object is a list containing:

B Matrix of basis functions evaluated at the training inputs.

vars, degs Arrays encoding the variables and polynomial degrees for each basis function across MCMC iterations.

nint, dtot Interaction order and total degree for each basis function.

nbasis Number of basis functions per MCMC iteration.

beta Posterior samples of regression coefficients.

s2 Posterior samples of the noise variance.

lam Posterior samples of the Poisson rate parameter controlling model size.

sum_sq Residual sum of squares at each iteration.

eta Variable inclusion counts used in the proposal distribution.

count_accept, count_propose Acceptance and proposal counts for RJMCMC moves.

prior_type Character string indicating the ridge prior.

X, y Training data.

References

Francom, Devin, and Bruno Sanso. "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." Journal of Statistical Software 94.LA-UR-20-23587 (2020).

Nott, David J., Anthony YC Kuk, and Hiep Duc. "Efficient sampling schemes for Bayesian MARS models with many predictors." Statistics and Computing 15 (2005): 93-101.

Rumsey, K.N., Francom, D., Gibson, G., Derek Tucker, J. and Huerta, G., 2026. Bayesian Adaptive Polynomial Chaos Expansions. Stat, 15(1), p.e70151.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- adaptive_khaos(X, y)
```

`ordinal_khaos` *Ordinal Probit Regression with BA-PCE*

Description

The emulation approach of Francom et al. (2020) for BMARS, modified for polynomial chaos.

Usage

```
ordinal_khaos(
  X,
  y,
  degree = 15,
  order = 5,
  nmcmc = 10000,
  nburn = 9000,
  thin = 1,
  max_basis = 1000,
  tau2 = 10^5,
  sigma_thresh = 10,
  h1 = 4,
  h2 = 40/length(y),
```

```
  move_probs = rep(1/3, 3),
  coin_pars = list(function(j) 1/j, 1, 2, 3),
  degree_penalty = 0,
  verbose = TRUE
)
```

Arguments

X	A data frame or matrix of predictors scaled to be between 0 and 1
y	a response vector
degree	Maximum polynomial degree for each basis function.
order	Maximum order of interaction for each basis function.
nmcmc	Number of MCMC iterations.
nburn	Number of initial MCMC iterations to discard.
thin	Keep every thin samples.
max_basis	Maximum number of basis functions.
tau2	Prior variance for coefficients
sigma_thresh	Prior variance for the threshold parameters.
h1, h2	Shape/rate parameters for the Gamma prior on expected number of basis functions.
move_probs	A 3-vector with probabilities for (i) birth, (ii) death, and (iii) mutation.
coin_pars	A list of control parameters for coinflip proposal
degree_penalty	Increasing this value encourages lower order polynomial terms (0 is no penalization).
verbose	Logical. Should progress information be printed?

Details

Implements the RJMCMC algorithm described by Francom & Sanso (2020) for BMARS, modifying it for polynomial chaos basis functions. As an alternative the NKD procedure of Nott et al. (2005), we use a coinflipping procedure to identify useful variables. See writeup (coming soon) for details. The coin_pars argument is an ordered list with elements:

1. a function giving proposal probability for q_0, the expected order of interaction,
2. epsilon, the base weight for each variable (epsilon=Inf corresponds to uniform sampling),
3. alpha, the exponent-learning rate,
4. num_passes a numerical parameter only.

Other proposal notes: Expected order q0 is chosen from 1:order with weights `coin_pars[[1]](1:order)` Degree is chosen from q0:degree with weights `(1:(degree-q0+1))^(-degree_penalty)` and a random partition is created (see helper function).

Two types of change steps are possible (and equally likely) A re-ordering of the degrees A single variable is swapped out with another one

Value

An object of class `"ordinal_khaos"`. The object is a list containing:

B Matrix of basis functions evaluated at the training inputs.

vars, degs Arrays encoding the variables and polynomial degrees for each basis function across MCMC iterations.

nint, dtot Interaction order and total degree for each basis function.

nbasis Number of basis functions per MCMC iteration.

beta Posterior samples of regression coefficients.

thresh Posterior samples of the ordinal threshold parameters.

s2z Estimated latent variance across iterations.

lam Posterior samples of the Poisson rate parameter controlling model size.

eta Variable inclusion counts used in the proposal distribution.

count_accept, count_propose Acceptance and proposal counts for RJMCMC moves.

X, y Training data.

References

Francom, Devin, and Bruno Sanso. "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." Journal of Statistical Software 94.LA-UR-20-23587 (2020).

Nott, David J., Anthony YC Kuk, and Hiep Duc. "Efficient sampling schemes for Bayesian MARS models with many predictors." Statistics and Computing 15 (2005): 93-101.

Examples

```
set.seed(1)
X <- matrix(runif(200), ncol = 1)
y <- sample(1:4, 200, replace = TRUE)
fit <- ordinal_khaos(X, y, nmcmc = 2000, nburn = 1000, thin = 2, verbose = FALSE)
plot(fit)
```

---

plot.adaptive_khaos *Plot Method for class adaptive_khaos*

Description

See `adaptive_khaos()` for details.

Usage

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

Arguments

x	An object returned by the adaptive_khaos() function.
...	Additional arguments for plotting

Details

Plot function for adaptive_khaos object.

Value

No return value; called for its side effects (plotting).

References

Francom, Devin, and Bruno Sanso. "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." Journal of Statistical Software 94.LA-UR-20-23587 (2020).

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- adaptive_khaos(X, y)
plot(fit)
```

---

plot.ordinal_khaos *Plot Method for class* ordinal_khaos

Description

Produces a 2x2 diagnostic plot: (1) trace of the number of basis functions; (2) jittered predicted vs. actual classes with point size indicating posterior support; (3) posterior mean class probabilities on x$X (lines if 1D, otherwise barplot of averages); (4) histogram of latent means $f(x)$ with posterior-average thresholds overlaid.

Usage

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

Arguments

x	An object of class ordinal_khaos.
...	additional arguments passed to plot

Details

Panel (2) uses `predict(x, type="class", aggregate=FALSE)` to compute per-draw MAP classes, plotting the modal class vs. observed class; point size reflects the fraction of draws supporting the modal class. Panel (3) uses `predict(x, type="prob")` to plot posterior mean class probabilities. Panel (4) uses `predict(x, type="latent", aggregate=FALSE)` to pool latent means across draws and overlays posterior-average cutpoints (handles either stored $\gamma_1, \ldots, \gamma_{K-1}$ or $\gamma_2, \ldots, \gamma_{K-1}$ with $\gamma_1 \equiv 0$).

Value

Invisibly returns `NULL`.

Examples

```
set.seed(1)
X <- matrix(runif(200), ncol = 1)
y <- sample(1:4, 200, replace = TRUE)
fit <- ordinal_khaos(X, y, nmcmc = 2000, nburn = 1000, thin = 2, verbose = FALSE)
plot(fit)
```

`plot.sparse_khaos` *Plot Method for class sparse_khaos*

Description

See `sparse_khaos()` for details.

Usage

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

Arguments

`x`	An object returned by the `sparse_khaos()` function.
`...`	additional arguments passed to `plot`

Details

Plot function for sparse_khaos object.

Value

returns invisible(x)

References

Shao, Q., Younes, A., Fahs, M., & Mara, T. A. (2017). Bayesian sparse polynomial chaos expansion for global sensitivity analysis. Computer Methods in Applied Mechanics and Engineering, 318, 474-496.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- sparse_khaos(X, y)
plot(fit)
```

---

predict.adaptive_khaos

Predict Method for class adaptive_khaos

Description

See `adaptive_khaos()` for details.

Usage

```
## S3 method for class 'adaptive_khaos'
predict(
  object,
  newdata = NULL,
  mcmc.use = NULL,
  nugget = FALSE,
  nreps = 1,
  ...
)
```

Arguments

`object`	An object returned by the `adaptive_khaos()` function.
`newdata`	A dataframe of the same dimension as the training data.
`mcmc.use`	Which posterior samples should be used for prediction?
`nugget`	Logical. Should predictions include error?
`nreps`	How many predictions should be taken for each mcmc sample (ignored when `nugget = FALSE`).
`...`	Additional arguments to predict

Details

Predict function for adaptive_khaos object.

Value

A numeric matrix of posterior predictive samples, with rows corresponding to samples and columns corresponding to observations.

References

Francom, Devin, and Bruno Sanso. "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." Journal of Statistical Software 94.LA-UR-20-23587 (2020).

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- adaptive_khaos(X, y)
predict(fit)
```

---

predict.ordinal_khaos *Predict Method for class* ordinal_khaos

---

Description

Compute posterior predictive quantities for an ordinal probit BA-PCE model. For each selected MCMC draw, the method builds the polynomial chaos basis, forms the linear predictor $f(x) = B(x)\beta$, reconstructs the ordered cutpoints, and returns either category probabilities, MAP classes, or latent means. Results can be averaged across draws or returned per-draw.

Usage

```
## S3 method for class 'ordinal_khaos'
predict(
  object,
  newdata = NULL,
  mcmc.use = NULL,
  type = c("class", "prob", "latent"),
  aggregate = FALSE,
  ...
)
```

Arguments

object	An object of class ordinal_khaos returned by ordinal_khaos().
newdata	A data frame or matrix of predictors on the same scale and with the same columns as used for fitting. If NULL, uses object$X.

`mcmc.use`	Integer vector of posterior draw indices to use. If `NULL`, uses all stored draws (i.e., `seq_along(object$nbasis)`).
`type`	Character; one of `"prob"`, `"class"`, or `"latent"`. `"prob"` returns category probabilities $P(Y = k \mid x)$ for $k = 1, \ldots, K$. `"class"` returns MAP classes. `"latent"` returns the latent mean $f(x)$ (i.e., $B(x)\beta$) for each draw.
`aggregate`	Logical (default `FALSE`); if `TRUE`, averages results across the selected MCMC draws (probabilities are averaged; classes are aggregated by simple voting; latent means are averaged). If `FALSE`, returns per-draw arrays.
`...`	Unused; included for method compatibility.

Details

Let $Z = f_\beta(X) + \varepsilon$ with $\varepsilon \sim \mathcal{N}(0, 1)$ and ordered cutpoints $\gamma_0 = -\infty < \gamma_1 < \cdots < \gamma_{K-1} < \gamma_K = \infty$. For a new input x, the category probabilities are

$$P(Y = k \mid x) = \Phi(\gamma_k - f(x)) - \Phi(\gamma_{k-1} - f(x)).$$

The function reconstructs the full cutpoint vector internally. If the fitted object stored $\gamma_1, \ldots, \gamma_{K-1}$ then these are used directly; if it stored only $\gamma_2, \ldots, \gamma_{K-1}$ (with $\gamma_1 \equiv 0$) the missing fixed cutpoint is inserted automatically.

Value

If `type = "prob"` and `aggregate = TRUE`, an `nrow(newdata) x K` numeric matrix of posterior mean category probabilities. If `aggregate = FALSE`, a 3-D array of dimension `(#draws) x nrow(newdata) x K`.

If `type = "class"` and `aggregate = TRUE`, an integer vector of length `nrow(newdata)` with MAP classes aggregated by voting across draws. If `aggregate = FALSE`, a `(#draws) x nrow(newdata)` integer matrix of per-draw MAP classes.

If `type = "latent"`, returns the latent mean(s) $f(x)$: a numeric vector of length `nrow(newdata)` if `aggregate = TRUE`, or a `(#draws) x nrow(newdata)` numeric matrix otherwise.

A numeric matrix of posterior predictive samples, with rows corresponding to samples and columns corresponding to observations.

References

Hoff, Peter D. (2009). *A First Course in Bayesian Statistical Methods*. Springer. (See ordinal probit data augmentation.)

Francom, Devin, and Bruno Sanso (2020). "BASS: An R package for fitting and performing sensitivity analysis of Bayesian adaptive spline surfaces." *Journal of Statistical Software* 94. LA-UR-20-23587.

See Also

`ordinal_khaos` for model fitting.

Examples

```
# Toy example (X scaled to [0,1])
set.seed(1)
X <- lhs::maximinLHS(200, 3)
# Suppose y has K=4 ordered categories:
fit <- ordinal_khaos(X, sample(1:4, 200, replace=TRUE),
                     nmcmc = 2000, nburn = 1000, thin = 2, verbose = FALSE)

# Posterior mean probabilities for each class
p_hat <- predict(fit, newdata = X, type = "prob", aggregate = TRUE)

# MAP classes (aggregated across draws)
y_hat <- predict(fit, newdata = X, type = "class")

# Latent mean f(x)
f_hat <- predict(fit, newdata = X, type = "latent")
```

---

predict.sparse_khaos *Predict Method for class sparse_khaos*

Description

See `sparse_khaos()` for details.

Usage

```
## S3 method for class 'sparse_khaos'
predict(object, newdata = NULL, samples = 1000, nugget = FALSE, ...)
```

Arguments

`object`	An object returned by the `sparse_khaos()` function.
`newdata`	A dataframe of the same dimension as the training data.
`samples`	How many posterior samples should be taken at each test point? If 0 or FALSE, then the MAP estimate is returned.
`nugget`	logical; should predictions include error? If FALSE, predictions will be for mean.
`...`	Additional arguments to predict

Details

Predict function for sparse_khaos object.

Value

A numeric matrix of predictive samples (or a vector if `samples = FALSE`), with rows corresponding to samples and columns corresponding to observations.

References

Shao, Q., Younes, A., Fahs, M., & Mara, T. A. (2017). Bayesian sparse polynomial chaos expansion for global sensitivity analysis. Computer Methods in Applied Mechanics and Engineering, 318, 474-496.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- sparse_khaos(X, y)
predict(fit)
```

---

print.sparse_khaos *Print Method for class sparse_khaos*

Description

See `sparse_khaos()` for details.

Usage

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

Arguments

`x`	An object returned by the `sparse_khaos()` function.
`...`	additional arguments passed to or from other methods.

Details

Print function for sparse_khaos object.

Value

The input object `x`, invisibly.

References

Shao, Q., Younes, A., Fahs, M., & Mara, T. A. (2017). Bayesian sparse polynomial chaos expansion for global sensitivity analysis. Computer Methods in Applied Mechanics and Engineering, 318, 474-496.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- sparse_khaos(X, y)
print(fit)
```

---

sobol_khaos *Compute Sobol Sensitivity Indices for khaos Models*

Description

Computes first-order, total-effect, and full interaction Sobol indices from an adaptive KHAOS model fit. Returns MCMC samples of sensitivity indices, including all active interaction terms used in the model.

Usage

```
sobol_khaos(obj, plot_it = TRUE, samples = 1000)
```

Arguments

obj	An object of class "adaptive_khaos" returned by adaptive_khaos() or of class "sparse_khaos" returned by sparse_khaos().
plot_it	Logical; if TRUE, boxplots of the sensitivity indices are shown.
samples	How many samples from the posterior (only used for sparse class).

Value

A list with the following components:

S A data frame of Sobol indices for all main effects and interactions (MCMC samples).

T A data frame of total-effect Sobol indices (MCMC samples).

leftover A numeric vector of unexplained variance proportions (residuals).

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391*((x[1]-0.4)*(x[2]-0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)
fit <- adaptive_khaos(X, y)
sob <- sobol_khaos(fit)
head(sob$S)
```

`sparse_khaos` *Bayesian Sparse Polynomial Chaos Expansion*

Description

A Bayesian sparse polynomial chaos expansion (PCE) method based on the forward-selection framework of Shao et al. (2017), with optional enrichment strategies that adapt the candidate basis set across degree and interaction-order expansions.

Usage

```
sparse_khaos(
  X,
  y,
  degree = c(2, 2, 16),
  order = c(1, 1, 5),
  prior = c(1, 1, 1),
  lambda = 1,
  rho = 0,
  regularize = TRUE,
  max_basis = 1e+06,
  enrichment = "adaptive",
  adaptive_ladder = c(10000, 1e+05),
  verbose = TRUE
)
```

Arguments

`X`	A data frame or matrix of predictors scaled to lie between 0 and 1.
`y`	A numeric response vector of length `nrow(X)`.
`degree`	Integer vector of length 3 giving the polynomial-degree schedule: `c(d_init, d_increment, d_max)`.
`order`	Integer vector of length 3 giving the interaction-order schedule: `c(q_init, q_increment, q_max)`.
`prior`	Numeric vector of prior hyperparameters `c(n0, v0, s0)`, where `n0` controls shrinkage of the regression coefficients, `v0` is the degrees-of-freedom parameter for the inverse-gamma prior on σ^2, and `s0` is the scale parameter for that prior.
`lambda`	Nonnegative penalty parameter used in the KIC calculation. Larger values favor sparser models.
`rho`	Basis functions are discarded after partial-correlation screening if their squared partial correlation is smaller than `rho`.
`regularize`	Logical; if `TRUE`, a weighted LASSO pre-screen is used to reduce the candidate basis set before partial-correlation ranking.

`max_basis` Maximum number of candidate basis functions allowed in any fitting round. This acts as a hard cap on computational cost.

`enrichment` Either an integer in `0:4` specifying a fixed enrichment strategy, or the character string `"adaptive"`, which allows the strategy to become more restrictive when the next candidate set would otherwise be too large. See Details.

`adaptive_ladder` Numeric vector of candidate-size thresholds used only when `enrichment = "adaptive"`. The ladder should have length between 1 and 4, be sorted in increasing order, and satisfy `max(adaptive_ladder) <= max_basis`. Internally, the ladder is used to determine when the algorithm should fall back to a more restrictive enrichment strategy before attempting the next recursion step.

`verbose` Logical; if `TRUE`, progress messages are printed.

Details

For fixed maximum degree and interaction order, the method constructs a candidate library of polynomial basis functions, ranks them using marginal correlations and sequential partial correlations, and evaluates nested models using the Kashyap information criterion (KIC). The best-ranked model is retained at that stage.

If the selected model contains a basis function at the current maximum degree or interaction order, the algorithm attempts another fitting round with increased degree and/or interaction order. In later rounds, the candidate basis set can be rebuilt using one of several enrichment strategies rather than constructing the full basis library from scratch.

The `enrichment` argument controls how the candidate set is generated after the first fitting round:

`0` **(local enrichment)** Starts from the previously selected basis functions and applies local modifications to each term. These include deletion, simplification, complication, and promotion moves. This is typically the fastest strategy, but it is also the most local.

`1` **(active-variable rebuild)** Rebuilds the full candidate library using only variables that were active in the previously selected model. This is computationally efficient, but inactive variables cannot re-enter once dropped.

`2` **(full active rebuild plus sparse inactive enrichment)** Rebuilds the candidate library on the currently active variables, then enriches around the previously selected basis functions by allowing inactive variables to re-enter through complication and promotion moves. This is a compromise between speed and flexibility.

`3` **(full active rebuild plus inactive enrichment)** Rebuilds the candidate library on the active variables, then enriches that larger active-variable library by allowing inactive variables to re-enter through complication and promotion moves. This is more expansive than `enrichment = 2`, but can be substantially more expensive.

`4` **(full rebuild)** Rebuilds the full candidate library over all variables at each enrichment step. This is the most exhaustive strategy.

`"adaptive"` Begins with a relatively expansive enrichment strategy and, before the next recursion step, uses an upper bound on the candidate-set size to decide whether a more restrictive strategy should be used instead. This recovers the computational benefit of avoiding construction of candidate sets that are known in advance to be too large.

When `enrichment = "adaptive"`, the function uses `adaptive_ladder` together with an upper bound on the next candidate-set size. If the current enrichment choice appears too expensive, the algorithm steps down to a more restrictive enrichment rule before proceeding. If even the most restrictive admissible next step is predicted to exceed the allowed budget, the current fitted model is returned.

Strategies `2`, `3`, and `"adaptive"` are designed to address a limitation of active-variable-only rebuilds: variables that are inactive in one round are allowed to re-enter the model in later rounds.

Value

An object of class `"sparse_khaos"`. The object is a list containing:

B Matrix of selected basis functions evaluated at the training inputs.

nbasis Number of selected basis functions (including intercept).

vars Matrix encoding the polynomial multi-indices for each basis function.

beta_hat MAP estimates of the regression coefficients.

s2 Estimated noise variance.

X, y Training data (with `y` on the original scale).

mu_y, sigma_y Centering and scaling constants used for `y`.

BtB, BtBi Cross-product matrix and its inverse.

G Prior scaling factors for coefficients.

prior Prior hyperparameters.

KIC Kashyap information criterion values for candidate models.

References

Shao, Q., Younes, A., Fahs, M., and Mara, T. A. (2017). Bayesian sparse polynomial chaos expansion for global sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering*, **318**, 474–496.

Rumsey, K. N., Francom, D., Gibson, G., Tucker, J. D., and Huerta, G. (2026). Bayesian adaptive polynomial chaos expansions. *Stat*, **15**(1), e70151.

Examples

```
X <- lhs::maximinLHS(100, 2)
f <- function(x) 10.391 * ((x[1] - 0.4) * (x[2] - 0.6) + 0.36)
y <- apply(X, 1, f) + stats::rnorm(100, 0, 0.1)

fit <- sparse_khaos(X, y)
```

Index