Package 'SLGP'

May 7, 2026

Type Package

Title Spatial Logistic Gaussian Process for Field Density Estimation

Version 1.0.2

Maintainer Athénaïs Gautier <athenais.gautier@onera.fr>

Description Provides tools for conditional and spatially dependent density estimation using Spatial Logistic Gaussian Processes (SLGPs). The approach represents probability densities through finite-rank Gaussian process priors transformed via a spatial logistic density transformation, enabling flexible non-parametric modeling of heterogeneous data. Functionality includes density prediction, quantile and moment estimation, sampling methods, and preprocessing routines for basis functions. Applications arise in spatial statistics, machine learning, and uncertainty quantification.
The methodology builds on the framework of Leonard (1978) <doi:10.1111/j.2517-6161.1978.tb01655.x>, Lenk (1988) <doi:10.1080/01621459.1988.10478625>, Tokdar (2007) <doi:10.1198/106186007X210206>, Tokdar (2010) <doi:10.1214/10-BA605>, and is further aligned with recent developments in Bayesian non-parametric modelling: see Gautier (2023) <https://boristheses.unibe.ch/4377/>, and Gautier (2025) <doi:10.48550/arXiv.2110.02876>).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Biarch true

Depends R (>= 3.5.0), stats

Imports DiceDesign, methods, mvnfast, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), GoFKernel, rstantools

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), rstan (>= 2.18.1), StanHeaders (>= 2.21.0)

SystemRequirements GNU make

Suggests knitr, rmarkdown, tidyr, dplyr, ggplot2, ggpubr, viridis, MASS

VignetteBuilder knitr

NeedsCompilation yes

Author Athénaïs Gautier [aut, cre]

Repository CRAN

Date/Publication 2026-02-17 21:00:02 UTC

Contents

predictSLGP_cdf 2
predictSLGP_moments 3
predictSLGP_newNode 5
predictSLGP_quantiles 7
retrainSLGP 8
sampleSLGP 10
slgp 12
SLGP-class 14

Index **16**

`predictSLGP_cdf` *Predict cumulative distribution values at new locations using a SLGP model*

Description

Computes the posterior cumulative distribution function (CDF) values at specified covariate values using a fitted SLGP model.

Usage

```
predictSLGP_cdf(
  SLGPmodel,
  newNodes,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  discrete = FALSE
)
```

Arguments

`SLGPmodel`	An object of class `SLGP-class`.
`newNodes`	A data frame with covariate values where the SLGP should be evaluated.
`interpolateBasisFun`	Character string indicating the interpolation scheme for basis functions: one of `"nothing"`, `"NN"`, or `"WNN"` (default).

nIntegral	Number of integration points along the response axis.
nDiscret	Discretization resolution for interpolation (optional).
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame with newNodes and predicted CDF values, columns named cdf_1, cdf_2, ...

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
# Set input and output ranges manually (you can also use range(Boston$age), etc.)
range_x <- c(0, 100)
range_response <- c(0, 50)

#' #Create a SLGP model but don't fit it
modelPrior <- slgp(medv ~ age,        # Use a formula to specify response and covariates
                   data = Boston,     # Use the original Boston housing data
                   method = "none",    # No training
                   basisFunctionsUsed = "RFF",         # Random Fourier Features
                   sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                   predictorsLower = range_x[1],         # Lower bound for 'age'
                   predictorsUpper = range_x[2],         # Upper bound for 'age'
                   responseRange = range_response,       # Range for 'medv'
                   opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                        MatParam = 5/2), # Matern 5/2 kernel
                   seed = 1)                             # Reproducibility

#Let us make 3 draws from the prior
nrep <- 3
set.seed(8)
p <- ncol(modelPrior@coefficients)
modelPrior@coefficients <- matrix(rnorm(n=nrep*p), nrow=nrep)

# Where to predict the field of pdfs ?
dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
seq(range_response[1], range_response[2],, 101)))
colnames(dfGrid) <- c("age", "medv")
predPriorcdf <- predictSLGP_cdf(SLGPmodel=modelPrior,
                                newNodes = dfGrid)
```

---

predictSLGP_moments *Predict centered or uncentered moments at new locations from a SLGP model*

Description

Computes statistical moments (e.g., mean, variance, ...) of the posterior predictive distributions at new covariate locations, using a given SLGP model.

Usage

```
predictSLGP_moments(
  SLGPmodel,
  newNodes,
  power,
  centered = FALSE,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  discrete = FALSE
)
```

Arguments

SLGPmodel	An object of class SLGP-class.
newNodes	A data frame of new covariate values.
power	Scalar or vector of positive integers indicating the moment orders to compute.
centered	Logical; if TRUE, computes centered moments. If FALSE, computes raw moments.
interpolateBasisFun	Interpolation mode for basis functions: "nothing", "NN", or "WNN" (default).
nIntegral	Number of integration points for computing densities.
nDiscret	Discretization resolution of the response space.
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame with:

- Repeated rows of the input covariates,
- A column power indicating the moment order,
- One or more columns mSLGP_1, mSLGP_2, ... for the estimated moments across posterior samples.

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
# Set input and output ranges manually (you can also use range(Boston$age), etc.)
range_x <- c(0, 100)
```

```
range_response <- c(0, 50)

# Train an SLGP model using Laplace estimation and RFF basis
modelLaplace <- slgp(medv ~ age,       # Use a formula to specify response and covariates
                 data = Boston,     # Use the original Boston housing data
                 method = "Laplace",    # Train using Maximum A Posteriori estimation
                 basisFunctionsUsed = "RFF",         # Random Fourier Features
                 sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                 predictorsLower = range_x[1],         # Lower bound for 'age'
                 predictorsUpper = range_x[2],         # Upper bound for 'age'
                 responseRange = range_response,       # Range for 'medv'
                 opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                      MatParam = 5/2), # Matern 5/2 kernel
                 seed = 1)                             # Reproducibility
dfX <- data.frame(age=seq(range_x[1], range_x[2], 1))
predMean <- predictSLGP_moments(SLGPmodel=modelLaplace,
                                newNodes = dfX,
                                power=c(1, 2),
                                centered=FALSE) # Uncentered moments of order 1 and 2
predVar <- predictSLGP_moments(SLGPmodel=modelLaplace,
                               newNodes = dfX,
                               power=c(2),
                               centered=TRUE) # Centered moments of order 2 (Variance)
```

`predictSLGP_newNode` *Predict densities at new covariate locations using a given SLGP model*

Description

Computes the posterior predictive probability densities at new covariate points using a fitted Spatial Logistic Gaussian Process (SLGP) model.

Usage

```
predictSLGP_newNode(
  SLGPmodel,
  newNodes,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  normalise = TRUE,
  discrete = TRUE
)
```

Arguments

`SLGPmodel`	An object of class `SLGP-class`.
`newNodes`	A data frame containing new covariate values at which to evaluate the SLGP.

interpolateBasisFun
: Character string indicating how basis functions are evaluated: one of "nothing", "NN", or "WNN" (default).

nIntegral
: Integer specifying the number of quadrature points over the response space.

nDiscret
: Integer specifying the discretization step for interpolation (only used if applicable).

normalise
: Boolean, indicates if we return normalised or unnormalised pdfs. (defaults to TRUE)

discrete
: Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame combining newNodes with columns named pdf_1, pdf_2, ..., representing the posterior predictive density for each sample of the SLGP.

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
# Set input and output ranges manually (you can also use range(Boston$age), etc.)
range_x <- c(0, 100)
range_response <- c(0, 50)

#' #Create a SLGP model but don't fit it
modelPrior <- slgp(medv ~ age,        # Use a formula to specify response and covariates
                data = Boston,     # Use the original Boston housing data
                method = "none",    # No training
                basisFunctionsUsed = "RFF",         # Random Fourier Features
                sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                predictorsLower = range_x[1],         # Lower bound for 'age'
                predictorsUpper = range_x[2],         # Upper bound for 'age'
                responseRange = range_response,       # Range for 'medv'
                opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                     MatParam = 5/2), # Matern 5/2 kernel
                seed = 1)                             # Reproducibility

#Let us make 3 draws from the prior
nrep <- 3
set.seed(8)
p <- ncol(modelPrior@coefficients)
modelPrior@coefficients <- matrix(rnorm(n=nrep*p), nrow=nrep)

# Where to predict the field of pdfs ?
dfGrid <- data.frame(expand.grid(seq(range_x[1], range_x[2], 5),
seq(range_response[1], range_response[2],, 101)))
colnames(dfGrid) <- c("age", "medv")
predPrior <- predictSLGP_newNode(SLGPmodel=modelPrior,
                                 newNodes = dfGrid)
```

predictSLGP_quantiles *Predict quantiles from a SLGP model at new locations*

Description

Computes quantile values at specified levels (probs) for new covariate points, based on the posterior CDFs from a trained SLGP model.

Usage

```
predictSLGP_quantiles(
  SLGPmodel,
  newNodes,
  probs,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  discrete = FALSE
)
```

Arguments

SLGPmodel	An object of class SLGP-class.
newNodes	A data frame of covariate values.
probs	Numeric vector of quantile levels to compute (e.g., 0.1, 0.5, 0.9).
interpolateBasisFun	Character string specifying interpolation scheme: "nothing", "NN", or "WNN" (default).
nIntegral	Number of integration points for computing the SLGP outputs.
nDiscret	Discretization level of the response axis (for CDF inversion).
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame with columns:

- The covariates in newNodes (repeated per quantile level),
- A column probs indicating the quantile level,
- Columns qSLGP_1, qSLGP_2, ... for each posterior sample's quantile estimate.

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
# Set input and output ranges manually (you can also use range(Boston$age), etc.)
range_x <- c(0, 100)
range_response <- c(0, 50)

# Train an SLGP model using Laplace estimation and RFF basis
modelLaplace <- slgp(medv ~ age,       # Use a formula to specify response and covariates
                data = Boston,     # Use the original Boston housing data
                method = "Laplace",    # Train using Maximum A Posteriori estimation
                basisFunctionsUsed = "RFF",         # Random Fourier Features
                sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                predictorsLower = range_x[1],         # Lower bound for 'age'
                predictorsUpper = range_x[2],         # Upper bound for 'age'
                responseRange = range_response,       # Range for 'medv'
                opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                     MatParam = 5/2), # Matern 5/2 kernel
                seed = 1)                             # Reproducibility
dfX <- data.frame(age=seq(range_x[1], range_x[2], 1))
# Predict some quantiles, for instance here the first quartile, median, third quartile
predQuartiles <- predictSLGP_quantiles(SLGPmodel= modelLaplace,
                                       newNodes = dfX,
                                       probs=c(0.25, 0.50, 0.75))
```

`retrainSLGP` *Retrain a fitted SLGP model with new data and/or estimation method*

Description

This function retrains an existing SLGP model using either a Bayesian MCMC estimation, a Maximum A Posteriori (MAP) estimation, or a Laplace approximation. The model can be retrained using new data, new inference settings, or updated hyperparameters. It reuses the structure and basis functions from the original model.

Usage

```
retrainSLGP(
  SLGPmodel,
  newdata = NULL,
  epsilonStart = NULL,
  method,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
```

```
  hyperparams = NULL,
  sigmaEstimationMethod = "none",
  seed = NULL,
  opts = list(),
  trend = NULL,
  verbose = FALSE
)
```

Arguments

SLGPmodel	An object of class SLGP-class to be retrained.
newdata	Optional data frame containing new observations. If NULL, the original data is reused.
epsilonStart	Optional numeric vector with initial values for the coefficients ϵ.
method	Character string specifying the estimation method: one of {"MCMC", "MAP", "Laplace"}.
interpolateBasisFun	Character string specifying how basis functions are evaluated: • "nothing" — evaluate directly at sample locations; • "NN" — interpolate using nearest neighbor; • "WNN" — interpolate using weighted nearest neighbors (default).
nIntegral	Integer specifying the number of quadrature points used to approximate integrals over the response domain.
nDiscret	Integer specifying the discretization grid size (used only if interpolation is enabled).
hyperparams	Optional list with updated hyperparameters. Must include: • sigma2: signal variance; • lengthscale: vector of lengthscales for the inputs.
sigmaEstimationMethod	Character string indicating how to estimate sigma2: either "none" (default) or "heuristic".
seed	Optional integer to set the random seed for reproducibility.
opts	Optional list of additional options passed to inference routines: stan_chains, stan_iter, ndraws, etc.
trend	Optional a function that returns the trend of the transformed GP (not to be estimated). If not provided, it defaults to 0.
verbose	Logical; if TRUE, print progress and diagnostic messages during computation. Defaults to FALSE.

Value

An updated object of class SLGP-class with retrained coefficients and updated posterior information.

References

Gautier, A. (2023). *Modelling and Predicting Distribution-Valued Fields with Applications to Inversion Under Uncertainty*. PhD Thesis, Universität Bern. https://boristheses.unibe.ch/4377/

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
range_x <- c(0, 100)
range_response <- c(0, 50)

#Create a SLGP model but don't fit it
modelPrior <- slgp(medv ~ age,        # Use a formula to specify response and covariates
                   data = Boston,     # Use the original Boston housing data
                   method = "none",    # No training
                   basisFunctionsUsed = "RFF",         # Random Fourier Features
                   sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                   predictorsLower = range_x[1],         # Lower bound for 'age'
                   predictorsUpper = range_x[2],         # Upper bound for 'age'
                   responseRange = range_response,       # Range for 'medv'
                   opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                        MatParam = 5/2), # Matern 5/2 kernel
                   seed = 1)                             # Reproducibility
#Retrain using the Boston Housing dataset and a Laplace approximation scheme
modelLaplace <- retrainSLGP(SLGPmodel=modelPrior,
                            newdata = Boston,
                            method="Laplace")
```

`sampleSLGP` *Draw posterior predictive samples from a SLGP model*

Description

Samples from the predictive distributions modeled by a SLGP at new covariate inputs. This method uses inverse transform sampling on the estimated posterior CDFs.

Usage

```
sampleSLGP(
  SLGPmodel,
  newX,
  n,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  seed = NULL,
```

```
  discrete = FALSE
)
```

Arguments

SLGPmodel	A trained SLGP model object (SLGP-class).
newX	A data frame of new covariate values at which to draw samples.
n	Integer or integer vector specifying how many samples to draw at each input point.
interpolateBasisFun	Character string specifying interpolation scheme for basis evaluation. One of "nothing", "NN", or "WNN" (default).
nIntegral	Integer; number of quadrature points for density approximation.
nDiscret	Integer; discretization step for the response axis.
seed	Optional integer to set a random seed for reproducibility.
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame containing sampled responses from the SLGP model, with covariate columns from newX and one response column named after SLGPmodel@responseName.

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
# Set input and output ranges manually (you can also use range(Boston$age), etc.)
range_x <- c(0, 100)
range_response <- c(0, 50)

# Train an SLGP model using Laplace estimation and RFF basis
modelMAP <- slgp(medv ~ age,        # Use a formula to specify response and covariates
                 data = Boston,     # Use the original Boston housing data
                 method = "MAP",    # Train using Maximum A Posteriori estimation
                 basisFunctionsUsed = "RFF",         # Random Fourier Features
                 sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                 predictorsLower = range_x[1],         # Lower bound for 'age'
                 predictorsUpper = range_x[2],         # Upper bound for 'age'
                 responseRange = range_response,       # Range for 'medv'
                 opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                      MatParam = 5/2), # Matern 5/2 kernel
                 seed = 1)                             # Reproducibility

# Let's draw new sample points from the SLGP

newDataPoints <- sampleSLGP(modelMAP,
                            newX = data.frame(age=c(0, 25, 95)),
```

```
                    n = c(10, 1000, 1), # how many samples to draw at each new x
                    interpolateBasisFun = "WNN")
```

---

slgp *Define and can train a Spatial Logistic Gaussian Process (SLGP) model*

Description

This function builds and trains an SLGP model based on a specified formula and data. The SLGP is a finite-rank Gaussian process model for conditional density estimation, trained using MAP, MCMC, Laplace approximation, or left untrained ("none").

Usage

```
slgp(
  formula,
  data,
  epsilonStart = NULL,
  method,
  basisFunctionsUsed,
  interpolateBasisFun = "NN",
  nIntegral = 101,
  nDiscret = 101,
  hyperparams = NULL,
  predictorsUpper = NULL,
  predictorsLower = NULL,
  responseRange = NULL,
  sigmaEstimationMethod = "none",
  seed = NULL,
  opts_BasisFun = list(),
  BasisFunParam = NULL,
  opts = list(),
  trend = NULL,
  verbose = FALSE
)
```

Arguments

formula	A formula specifying the model structure, with the response on the left-hand side and covariates on the right.
data	A data frame containing the variables used in the formula.
epsilonStart	Optional numeric vector of initial weights for the finite-rank GP: $Z(x,t) = \sum_{i=1}^{p} \epsilon_i f_i(x,t)$.
method	Character string specifying the training method: one of {"none", "MCMC", "MAP", "Laplace"}.

`basisFunctionsUsed`
: Character string describing the basis function type: one of "inducing points", "RFF", "Discrete FF", "filling FF", or "custom cosines".

`interpolateBasisFun`
: Character string indicating how to evaluate basis functions: "nothing" (exact eval), "NN" (nearest-neighbor), or "WNN" (weighted inverse-distance). Default is "NN".

`nIntegral`
: Number of quadrature points used for numerical integration over the response domain.

`nDiscret`
: Integer controlling the resolution of the interpolation grid (used only for "NN" or "WNN").

`hyperparams`
: Optional list of hyperparameters. Should contain:
  - `sigma2`: signal variance
  - `lengthscale`: vector of lengthscales (one per covariate)

`predictorsUpper`
: Optional numeric vector for the upper bounds of the covariates (used for scaling).

`predictorsLower`
: Optional numeric vector for the lower bounds of the covariates.

`responseRange`
: Optional numeric vector of length 2 with the lower and upper bounds of the response.

`sigmaEstimationMethod`
: Method to heuristically estimate the variance `sigma2`. Either "none" (default) or "heuristic".

`seed`
: Optional integer for reproducibility.

`opts_BasisFun`
: List of optional configuration parameters passed to the basis function initializer.

`BasisFunParam`
: Optional list of precomputed basis function parameters.

`opts`
: Optional list of extra settings passed to inference routines (e.g., `stan_iter`, `stan_chains`, `ndraws`).

`trend`
: Optional a function that returns the trend of the transformed GP (not to be estimated). If not provided, it defaults to 0.

`verbose`
: Logical; if `TRUE`, print progress and diagnostic messages during computation. Defaults to `FALSE`.

Value

An object of S4 class `SLGP-class`, containing:

coefficients Matrix of posterior (or prior) draws of the SLGP coefficients ϵ_i.

hyperparams List of fitted or provided hyperparameters.

logPost Log-posterior (if MAP or Laplace used).

method Estimation method used.

... Other internal information such as ranges, basis settings, and data.

References

Gautier, Athénaïs (2023). "Modelling and Predicting Distribution-Valued Fields with Applications to Inversion Under Uncertainty." Thesis, Universität Bern, Bern. https://boristheses.unibe.ch/4377/

Examples

```
# Load Boston housing dataset
library(MASS)
data("Boston")
# Set input and output ranges manually (you can also use range(Boston$age), etc.)
range_x <- c(0, 100)
range_response <- c(0, 50)

#' #Create a SLGP model but don't fit it
modelPrior <- slgp(medv ~ age,        # Use a formula to specify response and covariates
                   data = Boston,     # Use the original Boston housing data
                   method = "none",    # No training
                   basisFunctionsUsed = "RFF",         # Random Fourier Features
                   sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                   predictorsLower = range_x[1],         # Lower bound for 'age'
                   predictorsUpper = range_x[2],         # Upper bound for 'age'
                   responseRange = range_response,       # Range for 'medv'
                   opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                        MatParam = 5/2), # Matern 5/2 kernel
                   seed = 1)                             # Reproducibility

# Train an SLGP model using MAP estimation and RFF basis
modelMAP <- slgp(medv ~ age,        # Use a formula to specify response and covariates
                   data = Boston,     # Use the original Boston housing data
                   method = "MAP",    # Train using Maximum A Posteriori estimation
                   basisFunctionsUsed = "RFF",         # Random Fourier Features
                   sigmaEstimationMethod = "heuristic",  # Auto-tune sigma2 (more stable)
                   predictorsLower = range_x[1],         # Lower bound for 'age'
                   predictorsUpper = range_x[2],         # Upper bound for 'age'
                   responseRange = range_response,       # Range for 'medv'
                   opts_BasisFun = list(nFreq = 200,     # Use 200 Fourier features
                                        MatParam = 5/2),  # Matern 5/2 kernel
                   seed = 1)                             # Reproducibility
```

`SLGP-class` *The SLGP S4 Class: Spatial Logistic Gaussian Process Model*

Description

This S4 class represents a Spatial Logistic Gaussian Process (SLGP) model, designed for modeling conditional or spatially dependent probability distributions. It encapsulates all necessary components for training, sampling, and prediction, including the basis function setup, learned coefficients, and fitted hyperparameters.

Slots

`formula` A `formula` specifying the model structure and covariates.

`data` A `data.frame` containing the observations used to train the model.

`responseName` A `character` string specifying the name of the response variable.

`covariateName` A `character` vector specifying the names of the covariates.

`responseRange` A `numeric` vector of length 2 indicating the lower and upper bounds of the response.

`predictorsRange` A `list` containing:

- `predictorsLower`: lower bounds of the covariates;
- `predictorsUpper`: upper bounds of the covariates.

`method` A `character` string indicating the training method used: one of {"MCMC", "MAP", "Laplace", "none"}.

`p` An `integer` indicating the number of basis functions used.

`basisFunctionsUsed` A `character` string specifying the type of basis functions used: "inducing points", "RFF", "Discrete FF", "filling FF", or "custom cosines".

`opts_BasisFun` A `list` of additional options used to configure the basis functions.

`BasisFunParam` A `list` containing the computed parameters of the basis functions, e.g., Fourier frequencies or interpolation weights.

`coefficients` A `matrix` of coefficients for the finite-rank Gaussian process. Each row corresponds to a realization of the latent field: $Z(x,t) = \sum_{i=1}^{p} \epsilon_i f_i(x,t)$.

`hyperparams` A `list` of hyperparameters, including:

- `sigma`: numeric signal standard deviation;
- `lengthscale`: a vector of lengthscales for each input dimension.

`trend` A `function` that returns the trend of the transformed GP (not to be estimated).

`logPost` A `numeric` value representing the (unnormalized) log-posterior of the model. Currently available only for MAP and Laplace-trained models.

Index

predictSLGP_cdf, 2
predictSLGP_moments, 3
predictSLGP_newNode, 5
predictSLGP_quantiles, 7

retrainSLGP, 8

sampleSLGP, 10
SLGP *(*SLGP-class*)*, 14
slgp, 12
SLGP-class, 14