

Package ‘SLGP’

September 7, 2026

Type Package

Title Spatial Logistic Gaussian Process for Field Density Estimation

Version 1.1.1

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 2026-09-07 18:20:02 UTC

Contents

SLGP-package	2
coef,SLGP-method	4
formula,SLGP-method	4
nobs,SLGP-method	5
plot,SLGP,missing-method	5
predict,SLGP-method	7
predictSLGP_cdf	9
predictSLGP_moments	10
predictSLGP_newNode	11
predictSLGP_quantiles	12
print,SLGP-method	13
print.summary.SLGP	14
retrainSLGP	14
sampleSLGP	16
simulate,SLGP-method	17
slgp	18
SLGP-class	21
summary,SLGP-method	22
update,SLGP-method	22
Index	25

SLGP-package	<i>SLGP: A package for spatially dependent probability distributions</i>
--------------	--

Description

The **SLGP** package provides tools for fitting, summarising, predicting, simulating, updating, and visualising Spatial Logistic Gaussian Processes models (SLGP). SLGP models define flexible conditional distributions through finite-rank Gaussian process representations and can be fitted by MAP, Laplace approximation, or MCMC

Main user interface

The recommended workflow is based on standard R generics:

- `slgp`: fit or initialise an SLGP model.
- `summary`: summarise a fitted SLGP object.
- `plot`: visualise fitted conditional densities.
- `predict`: compute densities, CDFs, quantiles, or moments.
- `simulate`: draw conditional samples.
- `update`: refit an existing SLGP object.
- `coef`, `formula`, and `nobs`: standard extractors.

Low-level routines

The package also retains lower-level computational routines such as `predictSLGP_newNode`, `predictSLGP_cdf`, `predictSLGP_quantiles`, `predictSLGP_moments`, `sampleSLGP`, and `retrainSLGP`. These functions are primarily intended for advanced use and backward compatibility. For ordinary use, prefer the standard methods listed above.

Typical workflow

A typical analysis consists of:

1. Fitting an SLGP model with `slgp`.
2. Inspecting the fitted model with `summary` and `plot`.
3. Computing predictive quantities with `predict`.
4. Drawing samples from the predictive distributions with `simulate`.
5. Updating the fitted model with `update`, if required.

Author(s)

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

References

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

Examples

```
set.seed(1)
d <- data.frame(
  x = rep(seq(0, 1, length.out = 6), each = 5)
)
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

fit <- slgp(
```

```

y ~ x,
data = d,
method = "MAP",
basisFunctionsUsed = "RFF",
predictorsLower = 0,
predictorsUpper = 1,
responseRange = range(d$y),
opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),
seed = 1
)

summary(fit)

plot(fit, draw = c("mean", 1:5))

```

coef,SLGP-method	<i>Extract the coefficient draws of an SLGP model</i>
------------------	---

Description

Extract the coefficient draws of an SLGP model

Usage

```
## S4 method for signature 'SLGP'
coef(object, ...)
```

Arguments

object	An object of class SLGP-class .
...	Ignored.

Value

The matrix of finite-rank GP coefficients (draws in rows, basis functions in columns).

formula,SLGP-method	<i>Model formula of an SLGP object</i>
---------------------	--

Description

Model formula of an SLGP object

Usage

```
## S4 method for signature 'SLGP'
formula(x, ...)
```

Arguments

x An object of class [SLGP-class](#).
 ... Ignored.

Value

The model formula.

nobs, SLGP-method	<i>Number of observations used to fit an SLGP model</i>
-------------------	---

Description

Number of observations used to fit an SLGP model

Usage

```
## S4 method for signature 'SLGP'
nobs(object, ...)
```

Arguments

object An object of class [SLGP-class](#).
 ... Ignored.

Value

Integer number of observations.

plot,SLGP,missing-method	<i>Plot the estimated density field of an SLGP model</i>
--------------------------	--

Description

Visualises the conditional densities implied by a fitted (or prior) [SLGP-class](#) model. For a single covariate, the densities are drawn as response curves at a sequence of covariate slices.

Usage

```
## S4 method for signature 'SLGP,missing'
plot(
  x,
  y,
  newdata = NULL,
  n_slices = 6,
  n_response = 101,
  interpolateBasisFun = "WNN",
  draw = "mean",
  panels = TRUE,
  discrete = FALSE,
  ...
)
```

Arguments

<code>x</code>	An object of class SLGP-class .
<code>y</code>	Ignored (present for generic compatibility).#'
<code>newdata</code>	Optional data frame of covariate values at which to draw slices. If <code>NULL</code> , <code>n_slices</code> equally spaced covariate values spanning the model's predictor range are used.
<code>n_slices</code>	Number of covariate slices to display (single-covariate models only). Default 8.
<code>n_response</code>	Number of response grid points per slice. Default 101.
<code>interpolateBasisFun</code>	Integral-approximation scheme; default "WNN".
<code>draw</code>	Integer vector selecting coefficient draws to display, or "mean" to display the pointwise mean over all stored draws. Use <code>NULL</code> to display all available draws.
<code>panels</code>	Logical; if <code>TRUE</code> (default), draw one panel per covariate slice via <code>par(mfrow=)</code> . If <code>FALSE</code> , overlay all slices in a single plot.
<code>discrete</code>	Logical; whether the response is treated as discrete. Default <code>FALSE</code> .
<code>...</code>	Further graphical parameters passed to <code>matplot</code> .

Details

This method draws with base graphics so that the package gains a plot method without taking on a hard **ggplot2** dependency. It returns, the computed prediction `data.frame` invisibly, so users who prefer **ggplot2** can build their own figure from it.

Value

The prediction `data.frame`, invisibly.

Examples

```

set.seed(1)
d <- data.frame(x = rep(seq(0, 1, length.out = 6), each = 5))
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

fit <- slgp(
  y ~ x, data = d, method = "Laplace", basisFunctionsUsed = "RFF",
  predictorsLower = 0, predictorsUpper = 1, responseRange = range(d$y),
  opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),
  seed = 1, opts=list(ndraws=5)
)

plot(fit, draw = "mean")
plot(fit, draw = c("mean", 1:5))

```

predict,SLGP-method	<i>Predict from a fitted SLGP model</i>
---------------------	---

Description

Single entry point for all distributional summaries produced by an [SLGP-class](#) model. The type argument selects what is returned and dispatches to the corresponding workhorse function:

"density" conditional density values, newdata must contain both the response and the covariate columns.

"cdf" conditional CDF values, newdata as above.

"quantiles" conditional quantiles at probabilities, newdata contains covariate columns only.

"moments" conditional moments of order(s) power, newdata contains covariate columns only.

Usage

```

## S4 method for signature 'SLGP'
predict(
  object,
  newdata,
  type = c("density", "cdf", "quantiles", "moments"),
  probs = NULL,
  power = NULL,
  centered = FALSE,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  discrete = FALSE,
  ...
)

```

Arguments

<code>object</code>	An object of class <code>SLGP-class</code> .
<code>newdata</code>	A <code>data.frame</code> of evaluation points. For type <code>%in% c("density", "cdf")</code> it must contain the response and covariate columns; for type <code>%in% c("quantiles", "moments")</code> only the covariate columns are required.
<code>type</code>	Character; one of "density" (default), "cdf", "quantiles", "moments".
<code>probs</code>	Numeric vector of probabilities in (0, 1); required when type = "quantiles".
<code>power</code>	Numeric vector of moment orders; required when type = "moments".
<code>centered</code>	Logical; for type = "moments", whether moments are centered. Default FALSE.
<code>interpolateBasisFun</code>	Integral-approximation scheme, one of "nothing", "NN", "WNN" (default).
<code>nIntegral, nDiscret</code>	Integration / discretisation resolutions passed through to the workhorse functions.
<code>discrete</code>	Logical; treat the response as discrete. Default FALSE.
<code>...</code>	Ignored.

Value

A `data.frame` as returned by the corresponding `predictSLGP_*` function.

Examples

```
set.seed(1)
d <- data.frame(
  x = rep(seq(0, 1, length.out = 6), each = 5)
)
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

fit <- slgp(
  y ~ x,
  data = d,
  method = "MAP",
  basisFunctionsUsed = "RFF",
  predictorsLower = 0,
  predictorsUpper = 1,
  responseRange = range(d$y),
  opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),
  seed = 1
)

## Prediction grid for density and CDF evaluations:
grid <- expand.grid(
  y = seq(min(d$y), max(d$y), length.out = 100),
  x = c(0.25, 0.75)
)

## Predict conditional densities
```



```

pred_density <- predict(fit, newdata = grid, type = "density")

## Predict conditional cumulative distribution functions
pred_cdf <- predict(fit, newdata = grid, type = "cdf")

## Prediction locations for summaries depending only on the covariates
newx <- data.frame(x = c(0.25, 0.75))

## Predict conditional quantiles
pred_q <- predict(fit, newdata = newx, type = "quantiles",
                  probs = c(0.25, 0.5, 0.75))

## Predict the first two raw moments
pred_m <- predict(fit, newdata = newx, type = "moments", power = c(1, 2))

```

predictSLGP_cdf	<i>Predict cumulative distribution values at new locations using a SLGP model</i>
-----------------	---

Description

predictSLGP_cdf() is deprecated; use [predict](#)(object, type = "cdf") instead.

Computes posterior predictive cumulative distribution function values at specified response and covariate locations for a fitted Spatial Logistic Gaussian Process (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 containing both the response column and the covariate columns at which the CDF is 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	discretisation resolution for interpolation (optional).
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame combining newNodes with columns named cdf_1, cdf_2, ..., containing predictive CDF evaluations for each coefficient draw stored in SLGPmodel.

See Also

[predict](#) for the recommended user interface.

predictSLGP_moments	<i>Predict conditional moments</i>
---------------------	------------------------------------

Description

predictSLGP_moments() is deprecated; use [predict](#)(object, type = "moments") instead.

Computes raw or centered moments of the posterior predictive distributions at specified covariate locations.

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	discretisation resolution of the response space.
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame containing the covariates in newNodes, a column power, and columns named mSLGP_1, mSLGP_2, ..., containing predictive moments for each coefficient draw stored in SLGPmodel.

See Also

[predict](#) for the recommended user interface.

predictSLGP_newNode	<i>Predict densities at new covariate locations using a given SLGP model</i>
---------------------	--

Description

predictSLGP_newNode() is deprecated; use [predict](#)(object, type = "density") instead.

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,
  normalize = TRUE,
  discrete = FALSE
)
```

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 discretisation step for interpolation (only used if applicable).
normalize	Boolean, indicates if we return normalized or unnormalized 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 coefficient draw stored in SLGPmodel.

See Also

[predict](#) for the recommended user interface.

predictSLGP_quantiles *Predict conditional quantiles*

Description

predictSLGP_quantiles() is deprecated; use [predict](#)(object, type = "quantiles") instead.

Computes predictive quantiles at specified covariate locations by numerical inversion of the posterior predictive CDF.

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 containing the covariate columns at which quantiles are evaluated.
probs	Numeric vector of probabilities in (0, 1).
interpolateBasisFun	Character string specifying interpolation scheme: one of "nothing", "NN", or "WNN" (default).
nIntegral	Number of integration points for computing the SLGP outputs.
nDiscret	discretisation resolution used for CDF inversion.
discrete	Boolean, indicates if we work with continuous pdfs (default, FALSE) or discrete probabilities

Value

A data frame containing the covariates in newNodes, a column probs, and columns named qSLGP_1, qSLGP_2, ..., containing predictive quantiles for each coefficient draw stored in SLGPmodel.

See Also

[predict](#) for the recommended user interface.

print,SLGP-method	<i>Print a brief description of an SLGP model</i>
-------------------	---

Description

Compact, one-screen summary of a fitted (or prior) [SLGP-class](#) object: the model formula, the estimation method, the basis family and its rank, the response range, the data dimensions, and the key hyperparameters. No internal slots are dumped.

Usage

```
## S4 method for signature 'SLGP'
print(x, ...)

## S4 method for signature 'SLGP'
show(object)
```

Arguments

x	An object of class SLGP-class .
...	Ignored, for S3/S4 generic compatibility.
object	An object of class SLGP-class .

Value

x, invisibly.

Examples

```
set.seed(1)
d <- data.frame(
  x = rep(seq(0, 1, length.out = 6), each = 5)
)
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

fit <- slgp(
  y ~ x,
  data = d,
  method = "none",
  basisFunctionsUsed = "RFF",
  predictorsLower = 0,
  predictorsUpper = 1,
  responseRange = range(d$y),
  opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),
  seed = 1
```

```
)
fit
```

```
print.summary.SLGP      Print method for SLGP summaries
```

Description

Print method for SLGP summaries

Usage

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

Arguments

`x` An object of class "summary.SLGP".
`...` Ignored.

Value

`x`, invisibly.

```
retrainSLGP      Refit an SLGP model
```

Description

`retrainSLGP()` is deprecated; use [update](#)(object, ...) instead.

Re-estimates an existing SLGP model, optionally with new data, under a chosen estimation method while reusing the existing basis representation and model ranges.

Usage

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

```

hyperparams = NULL,
sigmaEstimationMethod = "none",
seed = NULL,
opts = list(),
trend = NULL,
discrete = FALSE,
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 training method: one of "none", "MCMC", "MAP", or "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 discretisation 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 function returning the trend of the transformed GP. If not provided, a zero trend is used.
discrete	Logical; whether the response is treated as discrete, defaults as FALSE.
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.

See Also

[update](#) for the recommended user interface.

sampleSLGP

Draw conditional samples

Description

sampleSLGP() is deprecated; use [simulate\(object\)](#) instead.

Draws samples from the posterior predictive distributions at specified covariate locations using inverse transform sampling based on the estimated predictive CDF.

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; discretisation 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 simulated responses, with the covariate columns from newX and one response column named after the response variable of SLGPmodel.

See Also

[simulate](#) for the recommended user interface.

simulate,SLGP-method	<i>Simulate responses from a fitted SLGP model</i>
----------------------	--

Description

This method provides the standard user interface for simulating from a fitted [SLGP-class](#) object.

Usage

```
## S4 method for signature 'SLGP'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  discrete = FALSE,
  ...
)
```

Arguments

object	An object of class SLGP-class .
nsim	Number of samples to draw at each covariate value. Default 1.
seed	Optional integer seed for reproducibility.
newdata	A <code>data.frame</code> of covariate values.
interpolateBasisFun	Integral-approximation scheme; default "WNN".
nIntegral, nDiscret	Integration / discretisation resolutions.
discrete	Logical; treat the response as discrete. Default FALSE.
...	Ignored.

Value

A `data.frame` of sampled responses with the covariate columns from `newdata`.

Examples

```

set.seed(1)
d <- data.frame(
  x = rep(seq(0, 1, length.out = 6), each = 5)
)
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

fit <- slgp(
  y ~ x,
  data = d,
  method = "MAP",
  basisFunctionsUsed = "RFF",
  predictorsLower = 0,
  predictorsUpper = 1,
  responseRange = range(d$y),
  opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),
  seed = 1
)

## Draw 10 samples from the conditional distributions at two locations
sim <- simulate(
  fit,
  newdata = data.frame(x = c(0.25, 0.75)),
  nsim = 10
)

head(sim)

```

slgp

Fit a Spatial Logistic Gaussian Process model

Description

Builds a finite-rank Spatial Logistic Gaussian Process (SLGP) model for conditional density estimation. The model can be fitted by MAP, MCMC, or Laplace approximation, or initialised without fitting by setting `method = "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,
discrete = FALSE,
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", or "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.

<code>sigmaEstimationMethod</code>	Method to heuristically estimate the variance <code>sigma2</code> . Either "none" (default) or "heuristic".
<code>seed</code>	Optional integer for reproducibility.
<code>opts_BasisFun</code>	List of optional configuration parameters passed to the basis function initializer.
<code>BasisFunParam</code>	Optional list of precomputed basis function parameters.
<code>opts</code>	Optional list of extra settings passed to inference routines (e.g., <code>stan_iter</code> , <code>stan_chains</code> , <code>ndraws</code>).
<code>trend</code>	Optional function returning the trend of the transformed GP. If not provided, a zero trend is used.
<code>discrete</code>	Logical; whether the response is treated as discrete, defaults as FALSE.
<code>verbose</code>	Logical; if TRUE, print progress and diagnostic messages during computation. Defaults to FALSE.

Value

An object of S4 class `SLGP-class`. Standard methods are available for fitted objects, including `summary`, `plot`, `predict`, `simulate`, `update`, `coef`, `formula`, and `nobs`.

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

```
set.seed(1)
d <- data.frame(
  x = rep(seq(0, 1, length.out = 6), each = 5)
)
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

fit <- slgp(
  y ~ x,
  data = d,
  method = "none",
  basisFunctionsUsed = "RFF",
  predictorsLower = 0,
  predictorsUpper = 1,
  responseRange = range(d$y),
  opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),
  seed = 1
)

fit
summary(fit)
```

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 modelling 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.

summary, SLGP-method	<i>Summarise a fitted SLGP model</i>
----------------------	--------------------------------------

Description

Extends `print` with fit diagnostics where available: the log-posterior at the mode (MAP / Laplace), the number of coefficient draws, and basic structural information. The returned object has class `"summary.SLGP"` and its own print method.

Usage

```
## S4 method for signature 'SLGP'
summary(object, ...)
```

Arguments

object	An object of class <code>SLGP-class</code> .
...	Ignored.

Value

An object of class `"summary.SLGP"` (a list), returned invisibly.

update, SLGP-method	<i>Re-fit an SLGP model under a new method or with new data</i>
---------------------	---

Description

Standard-generic front end to `retrainSLGP`: re-estimates an existing `SLGP-class` model, optionally with new data, under a chosen estimation method, reusing the existing basis and ranges.

Usage

```
## S4 method for signature 'SLGP'
update(
  object,
  method,
  newdata = NULL,
  epsilonStart = NULL,
  interpolateBasisFun = "WNN",
  nIntegral = 101,
  nDiscret = 101,
  discrete = FALSE,
  hyperparams = NULL,
  sigmaEstimationMethod = "none",
```

```

    seed = NULL,
    opts = list(),
    trend = NULL,
    verbose = FALSE,
    ...
)

```

Arguments

object	An object of class SLGP-class .
method	Estimation method: one of "MCMC", "MAP", "Laplace".
newdata	Optional new data.frame; if NULL, the model's stored data are reused.
epsilonStart	Optional initial coefficient values.
interpolateBasisFun	Integral-approximation scheme; default "WNN".
nIntegral, nDiscret	Integration / discretisation resolutions.
discrete	Logical; whether the response is treated as discrete. Default FALSE.
hyperparams	Optional list of hyperparameters; if NULL, those of object are reused.
sigmaEstimationMethod	Variance-selection method; default "none".
seed	Optional integer seed.
opts	List of method-specific options (e.g. stan_chains, stan_iter for MCMC, ndraws for Laplace).
trend	Optional trend function.
verbose	Logical; verbosity. Default FALSE.
...	Ignored.

Value

An updated object of class [SLGP-class](#).

See Also

[retrainSLGP](#) for the low-level routine.

Examples

```

set.seed(1)
d <- data.frame(
  x = rep(seq(0, 1, length.out = 6), each = 5)
)
d$y <- rnorm(nrow(d), mean = sin(2 * pi * d$x), sd = 0.2)

prior <- slgp(
  y ~ x,

```

```
data = d,  
method = "none",  
basisFunctionsUsed = "RFF",  
predictorsLower = 0,  
predictorsUpper = 1,  
responseRange = range(d$y),  
opts_BasisFun = list(nFreq = 20, MatParam = 5 / 2),  
seed = 1  
)  
  
## Refit the same model structure by MAP  
fit_map <- update(prior, method = "MAP")
```


Index

`coef`, [3](#), [20](#)
`coef`, SLGP-method, [4](#)

`formula`, [3](#), [20](#)
`formula`, SLGP-method, [4](#)

`nobs`, [3](#), [20](#)
`nobs`, SLGP-method, [5](#)

`plot`, [3](#), [20](#)
`plot`, SLGP,missing-method, [5](#)
`predict`, [3](#), [9–13](#), [20](#)
`predict`, SLGP-method, [7](#)
`predictSLGP_cdf`, [3](#), [9](#)
`predictSLGP_moments`, [3](#), [10](#)
`predictSLGP_newNode`, [3](#), [11](#)
`predictSLGP_quantiles`, [3](#), [12](#)
`print`, [22](#)
`print`, SLGP-method, [13](#)
`print.summary.SLGP`, [14](#)

`retrainSLGP`, [3](#), [14](#), [22](#), [23](#)

`sampleSLGP`, [3](#), [16](#)
`show`, SLGP-method (`print`, SLGP-method), [13](#)
`simulate`, [3](#), [16](#), [17](#), [20](#)
`simulate`, SLGP-method, [17](#)
SLGP (SLGP-class), [21](#)
`slgp`, [3](#), [18](#)
SLGP-class, [21](#)
SLGP-package, [2](#)
`summary`, [3](#), [20](#)
`summary`, SLGP-method, [22](#)

`update`, [3](#), [14](#), [16](#), [20](#)
`update`, SLGP-method, [22](#)