Package 'lgpr'

September 30, 2019

Title Longitudinal Gaussian Process Regression

Version 0.24.2

Description Implements interpretable nonparametric analysis and covariate selection for longitudinal data using additive Gaussian process regression. Includes specialized non-stationary disease effect modeling features for biomedical studies. Bayesian inference for model parameters is performed using Stan.

License GPL (>=3)

Depends methods,
R (>= 3.4.0),
Rcpp (>= 0.12.0)

Imports bayesplot (>= 1.6),
rstan (>= 2.18.1),
rstantools (>= 1.5.1),
MASS (>= 7.3-50),
stats (>= 3.4),
ggplot2 (>= 3.1.0)

LinkingTo StanHeaders (>= 2.18.0),
rstan (>= 2.18.1),
BH (>= 1.66.0),
Rcpp (>= 0.12.0),
RcppEigen (>= 0.3.3.3.0)

Encoding UTF-8

LazyData true

NeedsCompilation yes

SystemRequirements GNU make

RoxygenNote 6.1.1

Suggests knitr,
rmarkdown,
testthat,
covr

VignetteBuilder knitr

R topics documented:

`lgpr-package` *The 'lgpr' package.*

Description

Longitudinal Gaussian Process regression. The package features

- Additive Gaussian process modeling of longitudinal data
- Posterior inference of the model (hyper)parameters using Stan
- Computation of covariate relevances, i.e. how much each covariate explains the target variable
- Specialized modeling of a non-stationary disease effect
- Functions for visualizing longitudinal data, posterior samples and model predictions
- Gaussian, Poisson or Negative Binomial observation models

Basic usage

- See the main function `lgp` for creating and fitting additive longitudinal GP models.
- Predictions outside the data can be computed using the function `lgp_predict`.
- See documentation of the function `simulate_data` for generating artificial data.
- For visualizing the data and results, see for example the functions
  - `plot_data`
  - `plot_samples`
  - `plot_components`
  - `plot_posterior_y`
  - `plot_simdata`

Author(s)

Juho Timonen (first.last at aalto.fi)

References

1. Carpenter, B. et al. (2017). *Stan: A probabilistic programming language*. Journal of Statistical Software 76(1).
2. Gabry, J. and Goodrich, B. (2018). *rstantools: Tools for Developing R Packages Interfacing with 'Stan'*. R package version 1.5.1.
3. Gabry, J. and Mahr, T. (2018). *bayesplot: Plotting for Bayesian Models*. R package version 1.6.0.
4. Stan Development Team (2018). *RStan: the R interface to Stan*. R package version 2.17.4. http://mc-stan.org

---

`add_test_caseIDs` *Add case IDs to test data frame*

Description

Add case IDs to test data frame

Usage

```
add_test_caseIDs(X_test, X_data)
```

Arguments

`X_test`	test data frame
`X_data`	data frame

Value

Updated X_test data frame.

affected *Select the affected individuals*

Description

Select the affected individuals

Usage

```
affected(object, medians.return = FALSE, threshold = 0.5)
```

Arguments

object	An object of class lgpfit.
medians.return	Should the medians of beta parameters also be returned?
threshold	A value that the median of beta has to exceed

Value

A binary vector indicating the individuals for which the disease effect is inferred to exist.

assess_convergence *Assess convergence of the chains*

Description

Assess convergence of the chains

Usage

```
assess_convergence(fit, verbose = TRUE, recompute = F)
```

Arguments

fit	An (incomplete) object of class lgpfit.
verbose	should convergence info be printed?
recompute	Should the Rhat statistics be recomputed?

Value

Potential scale reduction factors (R_hat).

average_predictions *Average predictions over samples*

Description

Average predictions over samples

Usage

```
average_predictions(LIST)
```

Arguments

LIST	a list over samples

Value

a list

check_data *Validate the 'data' input to* lgp *and resolve covariate types*

Description

Validate the 'data' input to lgp and resolve covariate types

Usage

```
check_data(data, varInfo, verbose)
```

Arguments

data	the data frame that was passed to lgp
varInfo	variable type info
verbose	can this print some info?

Value

a list

check_formula *Validate the formula of* lgp

Description

Checks if the input 'formula' to lgp_model are valid with the given data

Usage

```
check_formula(formula, data)
```

Arguments

formula	the formula that was passed to lgp_model
data	the data frame that was passed to lgp_model

Value

nothing

check_hyperparameter_names *An error message for wrong hyperparameter naming*

Description

An error message for wrong hyperparameter naming

Usage

```
check_hyperparameter_names(dist, correct)
```

Arguments

dist	the distribution
correct	the allowed hyperparameter names

Value

nothing

compute_kernel_matrices *Evaluate kernel matrices for each component*

Description

Used by compute_predictions.

Usage

```
compute_kernel_matrices(X1, X2, kernel_info)
```

Arguments

X1	Covariate matrix of size n1 x sum(D).
X2	Covariate matrix of size n2 x sum(D).
kernel_info	A list of parameters and other kernel info.

Value

An array of size n1 x n2 x sum(D).

compute_K_beta *Compute the multiplier matrix K_beta (to eneable heterogeneous disease effect)*

Description

Compute the multiplier matrix K_beta (to eneable heterogeneous disease effect)

Usage

```
compute_K_beta(beta, row_to_caseID_1, row_to_caseID_2)
```

Arguments

beta	a row vector of length N_cases
row_to_caseID_1	mapping from row index to case ID
row_to_caseID_2	mapping from row index to case ID

Value

a matrix

compute_K_var_mask *Compute the variance mask kernel matrix*

Description

Compute the variance mask kernel matrix

Usage

```
compute_K_var_mask(disAge1, disAge2, vm_params, stp, nan_replace = 0)
```

Arguments

disAge1	disease-related age covariate vector of length n1
disAge2	disease-related age covariate vector of length n2
vm_params	vector of two mask function parameters
stp	input warping steepness
nan_replace	value to replace nans in disAge vectors

Value

a matrix of size n1 x n2

compute_lppd *Compute log-posterior predictive density at test points*

Description

Compute log-posterior predictive density at test points

Usage

```
compute_lppd(PRED, y_test)
```

Arguments

PRED	predictions
y_test	values of the response variable at the test points

Value

a matrix with size n_samples x n_data

compute_predicted_components *Compute component-wise predictions at test points*

Description

Used by compute_predictions.

Usage

```
compute_predicted_components(KK, KKs, KKss, y_data, sigma_n, DELTA)
```

Arguments

KK	Kernel matrices data vs. data.
KKs	Kernel matrices test vs. data.
KKss	Kernel matrices test vs. test.
y_data	Response variable.
sigma_n	Noise standard deviation parameter.
DELTA	Diagonal jitter that ensures pos. def. kernel.

Value

A list containing predicted means and variances.

compute_predictions *Compute component-wise predictions at test points*

Description

Used by lgp_predict.

Usage

```
compute_predictions(X_data, y_data, X_test, params, D, info, cnames, TSCL,
  handle_extra = "warning")
```

Arguments

X_data	Covariate matrix (data points).
y_data	Response variable (data points).
X_test	Covariate matrix (test points).
params	Kernel function and other hyperparameters
D	a vector of length 6
info	other model info
cnames	Names of the model components.
TSCL	time scaling function and its inverse
handle_extra	What to do if test data contains individuals that are not in the training data? Must be 'silent', 'warning' or 'error'.

Value

A list.

---

compute_relevances *Covariate and component relevance calculations*

Description

Covariate and component relevance calculations

Usage

```
compute_relevances(FFF, y_data, info, D, ell_smooth, x_age)
```

Arguments

FFF	a data frame of size n_data x n_components+2
y_data	(scaled) measurements of the response variable
info	model info
D	a vector of length 6
ell_smooth	lengthscale for kernel smoothing
x_age	(scaled) age covariate

Value

a list

create_covariates_stan *Create the covariate matrix that is given to stan*

Description

Create the covariate matrix that is given to stan

Usage

```
create_covariates_stan(data, varInfo, types, formula, verbose)
```

Arguments

data	the data frame that was passed to lgp
varInfo	original variable type info
types	the types returned by check_data
formula	the model formula
verbose	can this print some info?

Value

a list

create_data_plot_df *Create a plotting data frame for ggplot*

Description

A helper function for plot_data.

Usage

```
create_data_plot_df(data, hl_1, hl_2, hl_cont)
```

Arguments

data	a data frame
hl_1	highlighting by color
hl_2	highlighting by linestyle
hl_cont	highlighting continuous

Value

an extended data frame

create_F *Simulate latent function components for longitudinal data analysis*

Description

Simulate latent function components for longitudinal data analysis

Usage

```
create_F(X, covariates, relevances, lengthscales, X_affected, dis_fun,
  useBinKernel, steepness, vm_params)
```

Arguments

X	input data matrix (generated by create_X)
covariates	Integer vector that defines the types of covariates (other than id and age). Different integers correspond to the following covariate types: • 0 = disease-related age • 1 = other continuous covariate • 2 = a categorical covariate that interacts with age • 3 = a categorical covariate that acts as a group offset • 4 = a categorical covariate that that acts as a group offset AND is restricted to have value 0 for controls and 1 for cases
relevances	Relative relevance of each component. Must have be a vector so that length(relevances) = 2 + length(covariates). First two values define the relevance of the infividual-specific age and shared age component, respectively.
lengthscales	A vector so that length(lengthscales) = 2 + sum(covariates %in% c(0,1,2)).
X_affected	which individuals are affected by the disease
dis_fun	A function or a string that defines the disease effect. If this is a function, that function is used to generate the effect. If dis_fun is "gp_vm" or "gp_ns", the disease component is drawn from a nonstationary GP prior (vm is the variance masked version of it).
useBinKernel	Should the binary kernel be used for categorical covariates? If this is TRUE, the effect will exist only for group 1.
steepness	Steepness of the input warping function. This is only used if the disease component is in the model.
vm_params	Parameters of the variance mask function. This is only needed if useMaskedVarianceKernel = TRUE.

Value

a data frame FFF where one column corresponds to one additive data component

`create_predictions_plot_df1` *Create a plotting data frame for ggplot*

Description

A helper function for `plot_predictions`.

Usage

```
create_predictions_plot_df1(fit, scale_f = TRUE, n_sds)
```

Arguments

`fit`	An object of class `lgpfit`.
`scale_f`	Should the predictions be scaled back to the original data scale?
`n_sds`	number of standard deviations for the uncertainty band width

Value

a data frame

`create_predictions_plot_df2` *Create a plotting data frame for ggplot*

Description

A helper function for `plot_predictions`.

Usage

```
create_predictions_plot_df2(model, PRED, scale_f = TRUE,
  componentwise = FALSE, mode, n_sds)
```

Arguments

`model`	An object of class `lgpmodel`.
`PRED`	Predictions computed using `lgp_predict`.
`scale_f`	Should the predictions be scaled back to the original data scale?
`componentwise`	Should the predictions be plotted componentwise?
`mode`	mode
`n_sds`	number of standard deviations for the uncertainty band width

Value

a data frame

`create_simdata_plot_df` *Create a plotting data frame for ggplot*

Description

A helper function for `plot_simdata_by_component`.

Usage

```
create_simdata_plot_df(simData)
```

Arguments

`simData`	An object created using `simulate_data`.

Value

a data frame

`create_stan_input` *Create input for Stan*

Description

Parses the `formula` and `data` input to `lgp_model`. Also performs many input checks.

Usage

```
create_stan_input(formula, data, prior, likelihood, varInfo, standardize,
  uncertain_diagnosis, equal_effect, C_hat, DELTA, sample_F, t_test,
  verbose, variance_mask, cat_interact_kernel_type)
```

Arguments

`formula`	A formula of the form `y ~ x1 + x2 + x3` defining the response variable `y` and covariates `xi`. All variables that appear in the formula must exist as columns of `data`.
`data`	A data frame containing (at least) the variables given in `formula`.
`prior`	Prior distribution. Can be created for example using the function `prior_default`.
`likelihood`	Either `"Gaussian"` (default), `"Poisson"` or `"NB"`.

varInfo	Variable type info.
standardize	Should the response variable be standardized?
uncertain_diagnosis	Do we wish to model uncertainty in the disease effect time?
equal_effect	Is the disease effect assumed to be equally strong for all diseased individuals?
C_hat	This can only be given if likelihood is not Gaussian. The signal f will the be transformed so that g = exp(C_hat + f). If NULL, it will be set to C_hat = log(mean(y)), where y is the response variable.
DELTA	the amount of added jitter to ensure positive definiteness of the kernel
sample_F	Determines if the function values are be sampled (must be TRUE if likelihood is not Gaussian).
t_test	Optional test time points. Should only be used if sample_F = TRUE. Otherwise use lgp_predict after fitting the model.
verbose	Can this print some info?
variance_mask	Should a variance mask be used to force disease component variance to zero before disease onset?
cat_interact_kernel_type	Kernel type for categorical variables (other than id). Possible options are "categorical" (default) and "binary" (mask kernel where only category "1" will have an effect).

Value

A list containing the data to be given to rstan::sampling, some info about preprocessing and all the information about scaling the inputs and response, and updated variable type info.

---

create_test_points *Create a matrix of test points*

Description

Create a matrix of test points

Usage

```
create_test_points(object, t_test)
```

Arguments

object	An object of class lgpmodel or lgpfit
t_test	Test time points (will be same for each individual).

Value

A data frame.

create_X *Simulate an input data frame X*

Description

Simulate an input data frame X

Usage

```
create_X(N, covariates, names, n_categs, t_data, t_jitter, onset_range,
  continuous_info)
```

Arguments

N	Number of individuals.
covariates	Integer vector that defines the types of covariates (other than id and age). If not given, only the id and age covariates are created. Different integers correspond to the following covariate types: • 0 = disease-related age • 1 = other continuous covariate • 2 = a categorical covariate that interacts with age • 3 = a categorical covariate that acts as a group offset • 4 = a categorical covariate that that acts as a group offset AND is restricted to have value 0 for controls and 1 for cases
names	Covariate names.
n_categs	An integer vector defining the number of categories for each categorical covariate, so that `length(n_categs)` equals to the number of 2's and 3's in the `covariates` vector.
t_data	Measurement times.
t_jitter	Standard deviation of the jitter added to the given measurement times.
onset_range	Time interval from which the disease effect times are sampled uniformly. Alternatively, This can any function that returns the (possibly randomly generated) real disease effect time for one individual.
continuous_info	Info for generating continuous covariates. Must be a list containing fields `lambda` and `mu`, which have length 3. The continuous covariates are generated so that `x <- sin(a*t + b) + c`, where • `t <- seq(0, 2*pi, length.out = k)` • `a <- mu[1] + lambda[1]*stats::runif(1)` • `b <- mu[2] + lambda[2]*stats::runif(1)` • `c <- mu[3] + lambda[3]*stats::runif(1)`

Value

```
list(X, onsets, par_cont)
```

create_X_star *Create X_star*

Description

Create X_star

Usage

```
create_X_star(X, D, t_test, SCL, X_notnan)
```

Arguments

X	covariate matrix
D	covariate type information
t_test	Test time points (will be same for each individual).
SCL	time scaling function and its inverse
X_notnan	indicates where X_diseaseAge is not NaN

Value

A data frame.

create_y *Generate noisy observations*

Description

Generate noisy observations

Usage

```
create_y(noise_type, f, snr, phi)
```

Arguments

noise_type	Either "Gaussian", "Poisson" or "NB" (negative binomial).
f	The underlying signal.
snr	The desired signal-to-noise ratio. This argument is valid only with noise_type = "Gaussian".
phi	The dispersion parameter for negative binomial noise.

Value

A list `out`, where

- `out$g` is `f` mapped through an inverse link function and
- `out$y` is the noisy response variable.

`disease_effect` *Draw disease component from a parameteric form*

Description

Draw disease component from a parameteric form

Usage

```
disease_effect(X_id, X_disAge, dis_fun)
```

Arguments

`X_id`	the id covariate
`X_disAge`	the diseaseAge covariate
`dis_fun`	the disease age effect function

Value

a vector

`drawCategorical` *Indepedently draw categorical variables for each individual*

Description

Indepedently draw categorical variables for each individual

Usage

```
drawCategorical(N, k, v)
```

Arguments

`N`	number of individuals
`k`	number of timepoints
`v`	vector of numbers of different categories

Value

a matrix of size `N` x `D`, where `D <- length(v)`

drawContinuous *Indepedently draw continuous variables for each individual*

Description

Indepedently draw continuous variables for each individual

Usage

```
drawContinuous(N, k, D, mu, lambda)
```

Arguments

N	number of individuals
k	number of timepoints
D	number of variables
mu	a vector of length 3
lambda	a vector of length 3

Value

a matrix of size N x D

drawLatentComponents *Draw realizations of multivariate normals*

Description

Draw realizations of multivariate normals

Usage

```
drawLatentComponents(KK)
```

Arguments

KK	3D matrix where KK[,,j] is the jth kernel matrix

Value

a matrix FFF

drawMeasurementTimes *Draw the age covariate*

Description

Draw the age covariate

Usage

```
drawMeasurementTimes(N, t_data, t_jitter)
```

Arguments

N	number of individuals
t_data	a vector of length k
t_jitter	Standard deviation of the jitter added to the given measurement times.

Value

a vector of length N*k

extract_components_onesample *Extract inferred components for one sample*

Description

Extract inferred components for one sample

Usage

```
extract_components_onesample(fit, sample_idx)
```

Arguments

fit	an object of class lgpfit
sample_idx	sample index

Value

a list

extract_t_onset_samples *Extract samples of T_onset*

Description

Extract samples of T_onset

Usage

```
extract_t_onset_samples(fit)
```

Arguments

fit	an object of class lgpfit

Value

a matrix

get_case_ids *Get case ids in original data*

Description

Get case ids in original data

Usage

```
get_case_ids(fit)
```

Arguments

fit	an object of class lgpfit

Value

a character vector

get_case_row_mappings *Create case ID to rows and back mappings*

Description

Create mappings

- from case ID to data rows (caseID_to_rows, caseID_nrows)
- from row number to case ID (row_to_caseID)

Usage

```
get_case_row_mappings(X_notnan, X_id, only_R2C = FALSE)
```

Arguments

X_notnan	binary vector indicating if diseaseAge is available for that measurement
X_id	the id covariate in X
only_R2C	should this return only the rows-to-caseID mapping

Value

a list

get_diseased_info *Get some variables related to diseased individuals*

Description

Get some variables related to diseased individuals

Usage

```
get_diseased_info(D, X, X_notnan, uncertain_diagnosis, equal_effect, TSCL)
```

Arguments

D	an integer vector of length 6
X	the design matrix
X_notnan	a binary vector of length n
uncertain_diagnosis	Boolean value
equal_effect	Boolean value
TSCL	time scaling function and its inverse

Value

a list

`get_ell_smooth` *A convenience function used in postproc-main.R*

Description

A convenience function used in postproc-main.R

Usage

```
get_ell_smooth(ell_smooth, ell_smooth_multip, ell_smp)
```

Arguments

`ell_smooth`	a character or numeric argument
`ell_smooth_multip`	numeric
`ell_smp`	numeric

Value

a number

`get_function_component_samples` *Get values of sampled function components at data points*

Description

Get values of sampled function components at data points

Usage

```
get_function_component_samples(fit, only_at_datapoints)
```

Arguments

`fit`	An (incomplete) object of class `lgpfit`.
`only_at_datapoints`	Should the values be obtained only at data points or also test points?

Value

An array of size `n_samples` x `n_data` x `n_components+2` if `only_at_datapoints` is `TRUE`, else the size is `n_samples` x `n_total` x `n_components+2`

get_model_dims *Set a lot of generic variables that the Stan model needs as input*

Description

Set a lot of generic variables that the Stan model needs as input

Usage

```
get_model_dims(X, D, likelihood)
```

Arguments

X	the design matrix
D	a vector of length 6
likelihood	the ‘likelihood‘ input to lgp

Value

a list

get_onset_info *Get disease onset info*

Description

This returns

- a vector of observed onsets
- mapping from case ID to average sampling interval before the observed disease onset

Usage

```
get_onset_info(D, X, MAPS, TSCL)
```

Arguments

D	an integer vector of length 6
X	the design matrix
MAPS	mappings created by get_case_row_mappings
TSCL	time scaling function and its inverse

Value

two vectors of length N_cases

get_onset_times *Extract observed disease onset times from diseaseAge covariate vector*

Description

Extract observed disease onset times from diseaseAge covariate vector

Usage

```
get_onset_times(id, age, disAge)
```

Arguments

id	the id covariate, vector of length n
age	the age covariate, vector of length n
disAge	the observed disease-related age covariate, vector of length n

Value

vector of observed onset times

get_predicted *A helper function*

Description

A helper function

Usage

```
get_predicted(fit)
```

Arguments

fit	An (incomplete) object of class lgpfit.

Value

a list

get_prior_params *Get prior parameters*

Description

Get prior parameters

Usage

```
get_prior_params(dist, add_correct)
```

Arguments

dist	the distribution
add_correct	additional correct parameter names

Value

a hyperparameter vector of length 2

get_prior_type *A dictionary from distribution names to integer encoding*

Description

A dictionary from distribution names to integer encoding

Usage

```
get_prior_type(type)
```

Arguments

type	type of the distribution as a string

Value

an integer

get_response *Get the (scaled) response variable*

Description

Gets and possibly scales the response variable.

Usage

```
get_response(data, varInfo, standardize, likelihood)
```

Arguments

data	the data frame given as input to lgp
varInfo	variable type info
standardize	should the response be standardized to unit variance and zero mean
likelihood	the likelihood

Value

a list with the (scaled) response variable

get_runtime *Get average runtime of a chain*

Description

Get average runtime of a chain

Usage

```
get_runtime(object)
```

Arguments

object	An object of class lgpfit.

Value

Average runtimes for warmup and sampling

get_transform_type *A dictionary from transform names to integer encoding*

Description

A dictionary from transform names to integer encoding

Usage

```
get_transform_type(type)
```

Arguments

type	Type of the transform as a string. Allowed arguments are "none" or "square". If NULL, "none" is used.

Value

an integer (0, 1 or 2)

hyperparam_estimate *Get a posterior estimate of model (hyper)parameters*

Description

Get a posterior estimate of model (hyper)parameters

Usage

```
hyperparam_estimate(object, type = "mean")
```

Arguments

object	An (incomplete) object of class `lgpfit`.
type	Must be "mean", "median", or "map".

Value

a data frame

hyperparam_samples *Get a set of model (hyper)parameter samples*

Description

Get a set of model (hyper)parameter samples

Usage

```
hyperparam_samples(object, samples = NULL)
```

Arguments

object	An (incomplete) object of class lgpfit.
samples	Sample indices. If NULL, all samples are taken.

Value

a data frame

kernel_bin *Compute a binary kernel matrix*

Description

Compute a binary kernel matrix

Usage

```
kernel_bin(x1, x2 = NULL, alpha = 1, pos_class = 1)
```

Arguments

x1	(integer) vector of length n
x2	(integer) vector of length m
alpha	marginal std (default = 1)
pos_class	the positive class label

Value

A kernel matrix of size n x m

kernel_cat *Compute a categorical kernel matrix*

Description

Compute a categorical kernel matrix

Usage

```
kernel_cat(x1, x2, alpha = 1)
```

Arguments

x1	(integer) vector of length n
x2	(integer) vector of length m
alpha	marginal std (default = 1)

Value

A (binary) kernel matrix of size n x m

kernel_ns *Compute a nonstationary kernel matrix using input warping*

Description

Compute a nonstationary kernel matrix using input warping

Usage

```
kernel_ns(x1, x2 = NULL, alpha = 1, ell, a, b, c, nan_replace = 0)
```

Arguments

x1	vector of length n
x2	vector of length m
alpha	marginal std (default = 1)
ell	lengthscale in the warped space
a	steepness of the warping function rise
b	location of the effective time window
c	maximum range
nan_replace	the value to use for replacing NaN values

Value

A kernel matrix of size n x m

kernel_se *Compute a squared exponential kernel matrix*

Description

Compute a squared exponential kernel matrix

Usage

```
kernel_se(x1, x2, alpha = 1, ell = 1)
```

Arguments

x1	vector of length n
x2	vector of length m
alpha	marginal std (default = 1)
ell	lengthscale (default = 1)

Value

A kernel matrix of size n x m

kernel_smoothing *Estimate conditional mean time profile using gaussian kernel smoothing*

Description

Estimate conditional mean time profile using gaussian kernel smoothing

Usage

```
kernel_smoothing(v, t, t_out, ell)
```

Arguments

v	a vector of length n to be smoothed
t	vector of n time points corresponding to y
t_out	the set of p time points where the smoothing should be evaluated
ell	kernel lengthscale

Value

a vector of length p

lgp *The main function of the 'lgpr' package*

Description

This is a wrapper for both lgp_model and lgp_fit. It first creates an lgpmodel object and then fits the model, finally returning an lgpfit object.

Usage

```
lgp(formula, data, likelihood = "Gaussian", prior = prior_default(),
  uncertain_diagnosis = FALSE, equal_effect = TRUE,
  id_variable = "id", time_variable = "age", disAge_variable = NULL,
  continuous_vars = NULL, categorical_vars = NULL,
  offset_vars = NULL, C_hat = NULL, DELTA = 1e-12,
  sample_F = (likelihood != "Gaussian"), parallel = FALSE,
  skip_postproc = FALSE, t_test = NULL, threshold = 0.95,
  variance_mask = TRUE, ell_smooth = "ell_shared",
  ell_smooth_multip = 1, cat_interact_kernel_type = "categorical", ...)
```

Arguments

formula	A formula of the form `y ~ x1 + x2 + x3` defining the response variable y and covariates xi. All variables that appear in the formula must exist as columns of data.
data	A data frame containing (at least) the variables given in formula.
likelihood	Either "Gaussian" (default), "Poisson" or "NB".
prior	Prior distribution. Can be created for example using the function prior_default.
uncertain_diagnosis	Do we wish to model uncertainty in the disease effect time?
equal_effect	Is the disease effect assumed to be equally strong for all diseased individuals?
id_variable	Name of the unique subject identifier variable.
time_variable	Name of the time variable.
disAge_variable	Name of the disease-related age variable. If NULL, this will be chosen to be "diseaseAge", if such covariate is found in the data.
continuous_vars	Names of other continuous covariates. If NULL, the remaining covariates that have floating point values are interpreted as continuous.
categorical_vars	Names of categorical covariates that interact with the time variable. If NULL, the remaining covariates that have integer values are interpreted as categorical.
offset_vars	Names of the categorical covariates that are treated as time-independent group offsets. If NULL, no variables are interpreted as such covariates.

`C_hat` This can only be given if likelihood is not Gaussian. The signal `f` will the be transformed so that `g = exp(C_hat + f)`. If `NULL`, it will be set to `C_hat = log(mean(y))`, where `y` is the response variable.

`DELTA` the amount of added jitter to ensure positive definiteness of the kernel

`sample_F` Determines if the function values are be sampled (must be `TRUE` if likelihood is not Gaussian).

`parallel` Determines if the chain will be run in parallel (`default = FALSE`). If `TRUE`, then Stan is run by first defining `options(mc.cores = parallel::detectCores())`.

`skip_postproc` In this mode the postprocessing after running Stan is skipped.

`t_test` Optional test time points. Should only be used if `sample_F = TRUE`. Otherwise use `lgp_predict` after fitting the model.

`threshold` Covariate selection threshold.

`variance_mask` Should a variance mask be used to force disease component variance to zero before disease onset?

`ell_smooth` Defines how to determine smoothing lengthscale for corrected shared age effect inference. Possible options are

1. `"ell_shared"` (default) - the sampled lengthscale of the shared age component is used as `ell_smooth`
2. `"none"` - no correction will be performed
3. A numeric argument that directly defines `ell_smooth`

`ell_smooth_multip`
a multiplier for ell_smooth

`cat_interact_kernel_type`
Kernel type for categorical variables (other than id). Possible options are `"categorical"` (default) and `"binary"` (mask kernel where only category `"1"` will have an effect).

`...` Optional arguments passed to `rstan::sampling`, for example `iter`, `chains` or `control`. See `sampling` for the possible arguments.

Value

An object of class `lgpfit`.

`lgpfit-class` *An S4 class to represent the output of the* `lgp_fit` *function*

Description

All slots that are lists contain fields 'samples' and 'average'.

Slots

`stan_fit` The `stanfit` object returned by `rstan::sampling`.
`model` The `lgpmodel` object returned by `lgp_model`.
`components` Inferred components.
`components_corrected` Covariate-effect corrected components.
`component_relevances` Inferred component relevances.
`covariate_relevances` Inferred covariate relevances.
`covariate_selection` Covariate selection info.
`signal_variance` Signal variance.
`residual_variance` Residual variance.
`postproc_info` Postprocessing information.
`Rhat` Split Rhat statistics.

`lgpmodel-class` *An S4 class to represent an lgp model*

Description

An S4 class to represent an lgp model

Slots

`data` The original unmodified data frame.
`stan_dat` The data to be given as input to `rstan::sampling`.
`scalings` Preprocessing scaling functions and their inverse operations.
`info` Model info.

`lgp_component_names` *Get names of model components*

Description

Get names of model components

Usage

```
lgp_component_names(stan_dat)
```

Arguments

`stan_dat`	The data that was passed to `rstan::sampling`

Value

names of model components

lgp_covariate_names *Get names of model covariates*

Description

Get names of model covariates

Usage

```
lgp_covariate_names(stan_dat)
```

Arguments

stan_dat	The data that was passed to rstan::sampling

Value

names of model components

lgp_fit *Fit an lgp model*

Description

Samples the posterior of an additive Gaussian process regression model using rstan.

Usage

```
lgp_fit(model, threshold, parallel = FALSE, skip_postproc = FALSE,
  ell_smooth = "ell_shared", ell_smooth_multip = 1, ...)
```

Arguments

model	An object of class lgpmodel.
threshold	Covariate selection threshold.
parallel	Determines if the chain will be run in parallel (default = FALSE). If TRUE, then Stan is run by first defining options(mc.cores = parallel::detectCores()).
skip_postproc	In this mode the postprocessing after running Stan is skipped.
ell_smooth	Defines how to determine smoothing lengthscale for corrected shared age effect inference. Possible options are 1. "ell_shared" (default) - the sampled lengthscale of the shared age component is used as ell_smooth 2. "none" - no correction will be performed 3. A numeric argument that directly defines ell_smooth

ell_smooth_multip
: a multiplier for ell_smooth

...
: Optional arguments passed to rstan::sampling, for example iter, chains or control. See sampling for the possible arguments.

Value

An object of class lgpfit.

See Also

For the possible additional arguments, see sampling. For creating the lgpmodel input, see lgp_model.

---

lgp_model *Create an lgp model*

Description

Creates an object of class lgpmodel

Usage

```
lgp_model(formula, data, likelihood = "Gaussian",
  prior = prior_default(likelihood), uncertain_diagnosis = FALSE,
  equal_effect = TRUE, C_hat = NULL, DELTA = 1e-12,
  sample_F = (likelihood != "Gaussian"), t_test = NULL,
  id_variable = "id", time_variable = "age", disAge_variable = NULL,
  continuous_vars = NULL, categorical_vars = NULL,
  offset_vars = NULL, variance_mask = FALSE,
  cat_interact_kernel_type = "categorical")
```

Arguments

formula
: A formula of the form y ~ x1 + x2 + x3 defining the response variable y and covariates xi. All variables that appear in the formula must exist as columns of data.

data
: A data frame containing (at least) the variables given in formula.

likelihood
: Either "Gaussian" (default), "Poisson" or "NB".

prior
: Prior distribution. Can be created for example using the function prior_default.

uncertain_diagnosis
: Do we wish to model uncertainty in the disease effect time?

equal_effect
: Is the disease effect assumed to be equally strong for all diseased individuals?

C_hat
: This can only be given if likelihood is not Gaussian. The signal f will the be transformed so that g = exp(C_hat + f). If NULL, it will be set to C_hat = log(mean(y)), where y is the response variable.

DELTA	the amount of added jitter to ensure positive definiteness of the kernel
sample_F	Determines if the function values are be sampled (must be TRUE if likelihood is not Gaussian).
t_test	Optional test time points. Should only be used if sample_F = TRUE. Otherwise use lgp_predict after fitting the model.
id_variable	Name of the unique subject identifier variable.
time_variable	Name of the time variable.
disAge_variable	Name of the disease-related age variable. If NULL, this will be chosen to be "diseaseAge", if such covariate is found in the data.
continuous_vars	Names of other continuous covariates. If NULL, the remaining covariates that have floating point values are interpreted as continuous.
categorical_vars	Names of categorical covariates that interact with the time variable. If NULL, the remaining covariates that have integer values are interpreted as categorical.
offset_vars	Names of the categorical covariates that are treated as time-independent group offsets. If NULL, no variables are interpreted as such covariates.
variance_mask	Should a variance mask be used to force disease component variance to zero before disease onset?
cat_interact_kernel_type	Kernel type for categorical variables (other than id). Possible options are "categorical" (default) and "binary" (mask kernel where only category "1" will have an effect).

Value

An object of class lgpmodel.

See Also

For fitting the model, see lgp_fit.

---

lgp_predict *Compute predictions for a fitted model*

Description

Compute predictions for a fitted model. Only possible for models with Gaussian likelihood.

Usage

```
lgp_predict(fit, X_test, samples = "mean", print_progress = TRUE,
  print_params = FALSE)
```

Arguments

fit	An object of class lgpfit.
X_test	The test points where the predictions should be computed.
samples	The predictions can be computed either by using only the posterior mean (samples="mean"), median (samples="median"), or MAP (samples="map") parameters, or for all parameter samples (samples="all"). This can also be a set of indices, for example samples=c(1:10) gives predictions for the parameter samples 1...10.
print_progress	Should progress be printed (if there is more than one sample)?
print_params	Should the parameter values be printed? (only works if samples is mean or median.)

Value

A list.

See Also

- For creating an lgpfit object, see lgp_fit.
- For creating an lgpmodel object, see lgp_model.

---

lgp_test *Compute predictions and log-posterior predictive density at test points*

Description

This is a convenience function that wraps lgp_predict, compute_lppd and plot_posterior_y.

Usage

```
lgp_test(fit, test_data, plot = FALSE, verbose = TRUE,
  samples = "mean")
```

Arguments

fit	an object of class lgpfit
test_data	a test data matrix
plot	should this return also a plot of the data and predictions?
verbose	Should this print progress?
samples	Sample indices or a keyword "mean", "median", "map", or "all".

Value

a ggplot object or lppd

likelihood_as_str *Convert the Stan likelihood encoding to a string*

Description

Convert the Stan likelihood encoding to a string

Usage

```
likelihood_as_str(LH)
```

Arguments

LH	an integer

Value

a string

log_gaussian_density *Compute log-density for gaussian distribution*

Description

Compute log-density for gaussian distribution

Usage

```
log_gaussian_density(x, mu, s2)
```

Arguments

x	point x
mu	mean
s2	variance

Value

a number

matrix_to_df *Matrix to data frame without editing column names*

Description

Matrix to data frame without editing column names

Usage

```
matrix_to_df(M)
```

Arguments

M	a matrix

Value

a data frame

model_info *Get model info*

Description

Get model info

Usage

```
model_info(object, print = TRUE)
```

Arguments

object	an object of class `lgpmodel` or `lgpfit`
print	should this print the info?

Value

the info as a string

nameComponents *Create names for all components based on covariate names and types*

Description

Create names for all components based on covariate names and types

Usage

```
nameComponents(types, names)
```

Arguments

types	vector of covariate types
names	names of the covariates

Value

a vector of component names

onsetsToDiseaseAge *Compute the disease-related ages*

Description

Compute the disease-related ages

Usage

```
onsetsToDiseaseAge(onsets, age, k)
```

Arguments

onsets	true disease effect times, a vector of length N
age	the age covariate, a vector of length N*k
k	number of measurements per individual

Value

the diseaseAge covariate, a vector of length N*k

parse_prior_distribution

Turn a list describing a prior distribution into vectors to be given to Stan

Description

Turn a list describing a prior distribution into vectors to be given to Stan

Usage

```
parse_prior_distribution(dist, add_correct = NULL)
```

Arguments

dist	a list with field type, and possibly others
add_correct	additional correct parameter names

Value

a list with two vectors to be given to Stan

parse_prior_onset

Turn a list describing an onset prior distribution into things to be given to Stan

Description

Turn a list describing an onset prior distribution into things to be given to Stan

Usage

```
parse_prior_onset(dist, N_cases, T_observed, T_last, UNCRT)
```

Arguments

dist	This is prior$onset, where prior is an argument of lgp_model
N_cases	number of case individuals
T_observed	observed disease onsets
T_last	last time point for each diseased individual
UNCRT	0 or 1

Value

a list with things to be given to Stan

plot,lgpfit,ANY-method *Visualize a fitted 'lgpfit' object*

Description

Visualize a fitted 'lgpfit' object

Usage

```
## S4 method for signature 'lgpfit,ANY'
plot(fit, x = 1, y = 1, color_scheme = "red")
```

Arguments

fit	an object of class lgpfit
x	does nothing
y	does nothing
color_scheme	bayesplot color scheme

Value

a ggplot object

plot_beta *Visualize posterior samples of individual-specific disease effect magnitude parameters*

Description

Can only be used if the disease effect was modeled heterogeneously.

Usage

```
plot_beta(fit, color_scheme = "red", threshold = 0.5)
```

Arguments

fit	An object of class lgpfit.
color_scheme	Name of bayesplot color scheme.
threshold	Threshold for median.

Value

a ggplot object

plot_components *Visualize the (average) inferred components evaluated at data points*

Description

Visualize the (average) inferred components evaluated at data points

Usage

```
plot_components(fit, corrected = TRUE, title = NULL,
  sample_idx = NULL, linealpha = 0.6)
```

Arguments

fit	An object of class lgpfit.
corrected	Should this plot the covariate-effect corrected components?
title	optional prefix to plot title
sample_idx	If given, only one sample will be plotted, else the average components over all samples.
linealpha	line alpha

Value

a ggplot object

plot_data *A spaghetti plot of longitudinal data.*

Description

A spaghetti plot of longitudinal data.

Usage

```
plot_data(data, highlight = NULL, response = "y", id_variable = "id",
  time_variable = "age", psize = 2, lwd = 0.5, title = NULL)
```

Arguments

data	A data frame.
highlight	Name of a covariate to be highlighted with color, or id of a subject to be highlighted.
response	Name of the response variable.
id_variable	Name of id variable.
time_variable	Name of time variable.
psize	point size
lwd	line width
title	additional string added to title

Value

a ggplot object

---

plot_data_hl_cat *A spaghetti plot of longitudinal data, highlighting a categorical covariate.*

Description

A spaghetti plot of longitudinal data, highlighting a categorical covariate.

Usage

```
plot_data_hl_cat(data, highlight = NULL, response = "y",
  id_variable = "id", time_variable = "age", psize = 2, lwd = 0.5)
```

Arguments

data	A data frame.
highlight	Name of a categorical covariate to be highlighted with color.
response	Name of the response variable.
id_variable	Name of id variable.
time_variable	Name of time variable.
psize	point size
lwd	line width

Value

a ggplot object

plot_data_hl_cont *A spaghetti plot of longitudinal data, highlighting a continuous covariate.*

Description

A spaghetti plot of longitudinal data, highlighting a continuous covariate.

Usage

```
plot_data_hl_cont(data, highlight = NULL, response = "y",
  id_variable = "id", time_variable = "age", psize = 2, lwd = 0.5,
  colgrad = ggplot2::scale_colour_gradient2())
```

Arguments

data	A data frame.
highlight	Name of a continuous covariate to be highlighted with color.
response	Name of the response variable.
id_variable	Name of id variable.
time_variable	Name of time variable.
psize	point size
lwd	line width
colgrad	color gradient

Value

a ggplot object

plot_data_hl_disease *A spaghetti plot of longitudinal data, highlighting based on disease group.*

Description

A spaghetti plot of longitudinal data, highlighting based on disease group.

Usage

```
plot_data_hl_disease(data, highlight = "diseaseAge", response = "y",
  id_variable = "id", time_variable = "age", psize = 2, lwd = 0.5)
```

Arguments

data	A data frame.
highlight	Name of the disease-related age variable.
response	Name of the response variable.
id_variable	Name of id variable.
time_variable	Name of time variable.
psize	point size
lwd	line width

Value

a ggplot object

---

plot_data_hl_individual

A spaghetti plot of longitudinal data, highlighting one individual.

Description

A spaghetti plot of longitudinal data, highlighting one individual.

Usage

```
plot_data_hl_individual(data, highlight = 1, response = "y",
  id_variable = "id", time_variable = "age", psize = 2, lwd = 0.5)
```

Arguments

data	A data frame.
highlight	Number indicating the individual to highlight.
response	Name of the response variable.
id_variable	Name of id variable.
time_variable	Name of time variable.
psize	point size
lwd	line width

Value

a ggplot object

plot_data_plain *A spaghetti plot of longitudinal data without highlighting.*

Description

A spaghetti plot of longitudinal data without highlighting.

Usage

```
plot_data_plain(data, response = "y", id_variable = "id",
  time_variable = "age", psize = 2, lwd = 0.5)
```

Arguments

data	A data frame.
response	Name of the response variable.
id_variable	Name of id variable.
time_variable	Name of time variable.
psize	point size
lwd	line width

Value

a ggplot object

plot_inputwarp *Visualize the input warping function for different parameter samples*

Description

Visualize the input warping function for different parameter samples

Usage

```
plot_inputwarp(fit, p = 300, color_scheme = "red", b = 0, c = 1)
```

Arguments

fit	An object of class lgpfit.
p	number of plot points
color_scheme	Name of bayesplot color scheme.
b	location of the effective time window (default = 0)
c	maximum range (default = 1)

Value

a ggplot object

`plot_onset` *Visualize posterior uncertainty in the disease effect times*

Description

Can only be used if the uncertainty of effect time was modeled.

Usage

```
plot_onset(fit, color_scheme = "red", prob = 1, prob_outer = 1,
  point_est = "none")
```

Arguments

`fit`	An object of class `lgpfit`.
`color_scheme`	Name of bayesplot color scheme.
`prob`	Inner interval
`prob_outer`	Outer interval
`point_est`	Point estimate type

Value

a ggplot object

`plot_posterior_components`
Plot posterior of the components of f

Description

Plot posterior of the components of f

Usage

```
plot_posterior_components(fit, PRED = NULL, color_scheme = "red",
  alpha = 0.1, alpha_line = 1, plot_uncertainty = TRUE,
  title = NULL, ylim = NULL, n_sds = 2, original_y_scale = FALSE)
```

Arguments

fit	An object of class lgpfit.
PRED	Predictions computed using lgp_predict.
color_scheme	Name of bayesplot color scheme or a list with fieds 'dark' and 'light'.
alpha	Ribbon fill opacity.
alpha_line	Line opacity
plot_uncertainty	Should an uncertainty ribbon be plotted?
title	optional prefix to plot title
ylim	y axis limits
n_sds	number of standard deviations for the uncertainty band width
original_y_scale	should the predictions be scaled back to original data scale

Value

a ggplot object

---

plot_posterior_f *Plot posterior of f*

Description

This is a wrapper for plot_posterior_components. and plot_posterior_predictions.

Usage

```
plot_posterior_f(fit, PRED = NULL, componentwise = FALSE,
  plot_uncertainty = TRUE, n_sds = 2)
```

Arguments

fit	An object of class lgpfit.
PRED	Predictions computed using lgp_predict.
componentwise	A boolean value.
plot_uncertainty	Should an uncertainty ribbon be plotted?
n_sds	number of standard deviations for the uncertainty band width

Value

a ggplot object

plot_posterior_predictions *Plot posterior of f or predictive distribution for y*

Description

Plot posterior of f or predictive distribution for y

Usage

```
plot_posterior_predictions(fit, mode, PRED = NULL,
  color_scheme = "red", color_scheme_onset = "gray", alpha = 0.5,
  alpha_line = 1, alpha2 = 0.5, plot_uncertainty = TRUE,
  title = NULL, ylim = NULL, plot_obs_onset = FALSE,
  plot_onset_samples = FALSE, ypos_dens = NULL, test_data = NULL,
  color_test = "deepskyblue2", pch_test = 21, size_test = 2,
  error_bar = FALSE, n_sds = 2, reference_onsets = NULL,
  post_onset_statistic = "none", original_y_scale = TRUE,
  data_color = "black", data_marker = 21, ons_linetypes = c(1, 2, 3),
  ons_linecolors = c("black", "red", "gray50"))
```

Arguments

fit	An object of class lgpfit.
mode	Must be either "posterior" or "predictive".
PRED	Predictions computed using lgp_predict.
color_scheme	Name of bayesplot color scheme or a list with fieds 'dark' and 'light'.
color_scheme_onset	color scheme name for effect time density plotting
alpha	Ribbon fill opacity.
alpha_line	Line opacity.
alpha2	alpha of t_onset density
plot_uncertainty	Should an uncertainty ribbon be plotted?
title	optional prefix to plot title
ylim	y axis limits
plot_obs_onset	should the observed disease onset/initiation time be plotted by a vertical line
plot_onset_samples	should a distribution of sampled effect times be plotted
ypos_dens	y-position of the density plot
test_data	Test data frame
color_test	test point color

pch_test	test point marker
size_test	test point size
error_bar	should uncertainty be plotted using error bars instead of a ribbon
n_sds	number of standard deviations for the uncertainty band width
reference_onsets	reference onset times
post_onset_statistic	statistic computed from effect time samples (mean or median)
original_y_scale	should the predictions be scaled back to original data scale
data_color	data marker color
data_marker	data marker type
ons_linetypes	onset line types
ons_linecolors	onset line colors

Value

a ggplot object

`plot_posterior_y` *Plot posterior predictive distribution*

Description

This is a wrapper for `plot_posterior_predictions`.

Usage

```
plot_posterior_y(fit, PRED, uncertainty = "ribbon", test_data = NULL,
  n_sds = 2)
```

Arguments

fit	An object of class `lgpfit`.
PRED	Predictions computed using `lgp_predict`.
uncertainty	Either "none", "ribbon" or "errorbar".
test_data	Test data set.
n_sds	number of standard deviations for the uncertainty band width

Value

a ggplot object

`plot_predictions_add_onsets` *Add disease onset / effect times to predictions plot*

Description

NOTE: currently assumes that diseased individuals come first.

Usage

```
plot_predictions_add_onsets(fit, h, plot_obs_onset, plot_onset_samples,
  idvar, timevar, ypos_dens, color_scheme_onset, reference_onsets,
  post_onset_statistic, linetypes = c(1, 2, 3), linecolors = c("black",
  "red", "gray50"), alpha2 = 1)
```

Arguments

`fit`	An object of class `lgpfit`.
`h`	a ggplot object
`plot_obs_onset`	a boolean value
`plot_onset_samples`	a boolean value
`idvar`	id variable name
`timevar`	time variable name
`ypos_dens`	y position of the estimated onset density
`color_scheme_onset`	color scheme
`reference_onsets`	reference onset times
`post_onset_statistic`	statistic computed from effect time samples
`linetypes`	onset line types
`linecolors`	onset line colors
`alpha2`	alpha parameter

Value

a modified ggplot object

plot_predictions_options *Do input checks and set options for plotting predictions*

Description

Do input checks and set options for plotting predictions

Usage

```
plot_predictions_options(fit, color_scheme, componentwise,
  original_y_scale, PRED, test_data, color_scheme_onset, mode, n_sds)
```

Arguments

fit	An object of class lgpfit.
color_scheme	Name of bayesplot color scheme.
componentwise	Should the predictions be plotted componentwise?
original_y_scale	Boolean value.
PRED	Predictions computed using lgp_predict.
test_data	test data
color_scheme_onset	Another color scheme.
mode	mode
n_sds	number of standard deviations for the uncertainty band width

Value

a list

plot_relevances *Barplot of covariate relevances*

Description

Barplot of covariate relevances

Usage

```
plot_relevances(object, color_scheme = "red")
```

Arguments

object	an object of class lgpfit
color_scheme	bayesplot color scheme name

Value

a ggplot object

---

plot_samples *Visualize the distribution of the model parameter samples*

Description

This is a wrapper for functions in the bayesplot package.

Usage

```
plot_samples(object, pars = character(), regex_pars = character(),
  type = "intervals", prob = 0.5, prob_outer = 0.9,
  color_scheme = "red", point_est = "median", binwidth = NULL,
  transformations = list(), off_diag_args = list(size = 1),
  facet_args = list())
```

Arguments

object	An object of class lgpfit.
pars	parameter names
regex_pars	regex for parameter names
type	Visualization type. Must be either "dens", "areas", "intervals"(default) or "hist".
prob	inner interval
prob_outer	outer interval
color_scheme	See different color schemes in the bayesplot package.
point_est	the point estimate type
binwidth	width of histogram bins if type = "hist"
transformations	the parameter transformations
off_diag_args	Additional argument list for the pairs plot.
facet_args	additional facetting arguments

Value

a ggplot object

plot_simdata *Visualize simulated data*

Description

This is a wrapper for `plot_simdata_by_individual` and `plot_simdata_by_component`

Usage

```
plot_simdata(simData, componentwise = FALSE, nrow = NULL,
  ncol = NULL, i_test = NULL, color_test = "steelblue2")
```

Arguments

`simData`	a list returned by `simulate_data`
`componentwise`	should each component be plotted separately?
`nrow`	an argument for `ggplot2::facet_wrap`
`ncol`	an argument for `ggplot2::facet_wrap`
`i_test`	test point indices
`color_test`	test point color

Value

a ggplot object

plot_simdata_by_component *Plot each component of a simulated longitudinal data set separately*

Description

Plot each component of a simulated longitudinal data set separately

Usage

```
plot_simdata_by_component(simData, linecolor = "black", nrow = NULL,
  ncol = NULL, plot_point = TRUE, linealpha = 1)
```

Arguments

`simData`	a list returned by `simulate_data`
`linecolor`	line color
`nrow`	an argument for `ggplot2::facet_wrap`
`ncol`	an argument for `ggplot2::facet_wrap`
`plot_point`	should points be plotted also
`linealpha`	line alpha

Value

a ggplot object

`plot_simdata_by_individual` *Plot a simulated longitudinal data set for each individual separately*

Description

Plot a simulated longitudinal data set for each individual separately

Usage

```
plot_simdata_by_individual(simData, linecolor = "gray70", nrow = NULL,
  ncol = NULL, i_test = NULL, color_test = "steelblue2")
```

Arguments

`simData`	a list returned by `simulate_data`
`linecolor`	line color
`nrow`	an argument for `ggplot2::facet_wrap`
`ncol`	an argument for `ggplot2::facet_wrap`
`i_test`	test point indices
`color_test`	test point color

Value

a ggplot object

`postproc` *Finalize the lgpfit object after sampling*

Description

Creates the `lgpfit` slots

1. `components` - Inferred components.
2. `components_corrected` - Covariate-effect corrected components.
3. `component_relevances` - Inferred component relevances.
4. `covariate_relevances` - Inferred covariate relevances.
5. `signal_variance` - Signal variance.
6. `residual_variance` - Residual variance.
7. `covariate_selection` - Covariate selection info

all of which are lists that contain the fields `samples` and `average`.

Usage

```
postproc(fit, threshold = 0.95, ell_smooth = "ell_shared",
  ell_smooth_multip = 1, sample_idx = NULL,
  average_before_variance = FALSE)
```

Arguments

`fit`	An (incomplete) object of class `lgpfit`.
`threshold`	Covariate selection threshold.
`ell_smooth`	Defines how to determine smoothing lengthscale for corrected shared age effect inference. Possible options are 1. `"ell_shared"` (default) - the sampled lengthscale of the shared age component is used as `ell_smooth` 2. `"none"` - no correction will be performed 3. A numeric argument that directly defines `ell_smooth`
`ell_smooth_multip`	a multiplier for ell_smooth
`sample_idx`	If supplied, this just returns the inferred components for one sample.
`average_before_variance`	Should the variances be computed using average components?

Value

An updated object of class `lgpfit`.

`predict_preproc` *Preprocess some things before computing predictions*

Description

This is a helper function for `lgp_predict`.

Usage

```
predict_preproc(fit, X_test, samples)
```

Arguments

`fit`	An object of class `lgpfit`.
`X_test`	The test points where the predictions should be computed.
`samples`	The samples argument to `lgp_predict`

print_prior *Human-readable description of a specified prior*

Description

Print human-readable info about the prior specification that was used or will be used

Usage

```
print_prior(object)
```

Arguments

object	An object of class lgpfit or a valid prior argument for the ‘lgp‘ function.

Value

nothing

prior_default *Create the default prior*

Description

Create the default prior

Usage

```
prior_default(sigma_alpha = 1)
```

Arguments

sigma_alpha	Sigma parameter of the student-t distribution for all alpha.

Value

A list defining a valid prior argument for the lgp function.

prior_LonGP *Create a similar default prior as in* LonGP *(Cheng et. al, 2019)*

Description

Not recommended, because a lengthscale close to 0 is possible.

Usage

```
prior_LonGP()
```

Value

A list defining a valid `prior` argument for the `lgp_model` function.

prior_stan_to_readable *Human-readable information about the priors in the Stan data object*

Description

Human-readable information about the priors in the Stan data object

Usage

```
prior_stan_to_readable(stan_dat)
```

Arguments

`stan_dat`	The list that is passed as data to `rstan::sampling`.

Value

Info as a string.

prior_statement *Human-readable prior statement*

Description

Human-readable prior statement

Usage

```
prior_statement(parname, TYP, P, dist, row_change = TRUE)
```

Arguments

parname	parameter name
TYP	two integers
P	three real numbers
dist	list of distribution names
row_change	should a newline be last character?

Value

Sampling statement as a string.

prior_to_stan *Get priors as a format that can be input to Stan*

Description

Get priors as a format that can be input to Stan

Usage

```
prior_to_stan(D, prior, HMGNS, UNCRT, N_cases, T_observed, T_last)
```

Arguments

D	an integer vector of length 6
prior	The prior argument supplied to lgp().
HMGNS	Is diseaseAge assumed to have a homogenous effect (1) or not (0)?
UNCRT	Boolean value, is uncertainty of disease onset modeled?
N_cases	number of case individuals
T_observed	observed disease onsets
T_last	last time point for each diseased individual

Value

a list with all things related to priors that Stan needs

`repvec` *Repeat a vector as a rows of an array*

Description

Repeat a vector as a rows of an array

Usage

```
repvec(v, n)
```

Arguments

`v`	a vector of length `m`
`n`	number of times to repeat

Value

returns an array of size `n` x `m`

`rtgeom` *Sample from the 'truncated geometric' distribution*

Description

Sample from the 'truncated geometric' distribution

Usage

```
rtgeom(s, p, n = 1)
```

Arguments

`s`	an integer
`p`	a number between 0 and 1
`n`	number of samples

Value

an integer from the interval 1...n

scaleRelevances *Scale the effect sizes*

Description

Scale the effect sizes

Usage

```
scaleRelevances(FFF, relevances, force_zero_mean = TRUE, i_dis)
```

Arguments

FFF	matrix where one column corresponds to one additive data component
relevances	the desired variance of each component (column)
force_zero_mean	should each component be forced to have zero mean?
i_dis	index of a component for which the zero-mean forcing is skipped

Value

a new matrix FFF

separate_effects *Separate the covariate effects from an interaction components of a categorical covariate and age*

Description

Separate the covariate effects from an interaction components of a categorical covariate and age

Usage

```
separate_effects(f_post, t, D, ell, i_edit)
```

Arguments

f_post	a matrix of size n x sum(D)
t	vector of n time points corresponding to f_post
D	a vector of length 6
ell	kernel lengthscale
i_edit	Indices of columns whose effect should be moved to shared age.

Value

a corrected f_post

show,lgpfit-method *Show a summary of results of the* lgp *function*

Description

Show a summary of results of the lgp function

Usage

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

Arguments

object	an object of class lgpfit

Value

nothing

show,lgpmodel-method *Show a summary of an* lgpmodel

Description

Show a summary of an lgpmodel

Usage

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

Arguments

object	an object of class lgpmodel

Value

nothing

show_relevances *Print info about component and covariate relevances*

Description

Print info about component and covariate relevances

Usage

```
show_relevances(fit)
```

Arguments

fit	an object of class lgpfit

Value

nothing

simdata_colnames_pretty *Simulated data column names in a prettier form*

Description

Simulated data column names in a prettier form

Usage

```
simdata_colnames_pretty(cn)
```

Arguments

cn	column names

Value

names of model components

simulate_data *Generate an artificial longitudinal data set*

Description

Generate an artificial longitudinal data set.

Usage

```
simulate_data(N, t_data, covariates = c(), names = NULL,
  relevances = c(1, 1, rep(1, length(covariates))), n_categs = rep(2,
  sum(covariates %in% c(2, 3))), t_jitter = 0, lengthscales = rep(12,
  2 + sum(covariates %in% c(0, 1, 2))), f_var = 1,
  noise_type = "Gaussian", snr = 3, phi = 1,
  N_affected = round(N/2), onset_range = "auto",
  t_observed = "after_0", C_hat = 0, dis_fun = "gp_vm",
  useBinKernel = TRUE, steepness = 0.5, vm_params = c(0.025, 1),
  continuous_info = list(mu = c(pi/8, pi, -0.5), lambda = c(pi/8, pi,
  1)))
```

Arguments

N	Number of individuals.
t_data	Measurement times.
covariates	Integer vector that defines the types of covariates (other than id and age). If not given, only the id and age covariates are created. Different integers correspond to the following covariate types: • 0 = disease-related age • 1 = other continuous covariate • 2 = a categorical covariate that interacts with age • 3 = a categorical covariate that acts as a group offset • 4 = a categorical covariate that that acts as a group offset AND is restricted to have value 0 for controls and 1 for cases
names	Covariate names.
relevances	Relative relevance of each component. Must have be a vector so that `length(relevances) = 2 + length(covariates)`. First two values define the relevance of the infividual-specific age and shared age component, respectively.
n_categs	An integer vector defining the number of categories for each categorical covariate, so that `length(n_categs)` equals to the number of 2's and 3's in the `covariates` vector.
t_jitter	Standard deviation of the jitter added to the given measurement times.
lengthscales	A vector so that `length(lengthscales) = 2 + sum(covariates %in% c(0,1,2))`.

f_var — variance of f

noise_type — Either "Gaussian", "Poisson" or "NB" (negative binomial).

snr — The desired signal-to-noise ratio. This argument is valid only with `noise_type = "Gaussian"`.

phi — The dispersion parameter for negative binomial noise.

N_affected — Number of diseased individuals that are affected by the disease. This defaults to the number of diseased individuals. This argument can only be given if `covariates` contains a zero.

onset_range — Time interval from which the disease effect times are sampled uniformly. Alternatively, This can any function that returns the (possibly randomly generated) real disease effect time for one individual.

t_observed — Determines how the disease onset is observed. This can be any function that takes the real disease effect time as an argument and returns the (possibly randomly generated) observed onset/initiation time. Alternatively, this can be a string of the form `"after_n"` or `"random_p"` or `"exact"`.

C_hat — A constant added to f

dis_fun — A function or a string that defines the disease effect. If this is a function, that function is used to generate the effect. If `dis_fun` is "gp_vm" or "gp_ns", the disease component is drawn from a nonstationary GP prior (vm is the variance masked version of it).

useBinKernel — Should the binary kernel be used for categorical covariates? If this is `TRUE`, the effect will exist only for group 1.

steepness — Steepness of the input warping function. This is only used if the disease component is in the model.

vm_params — Parameters of the variance mask function. This is only needed if `useMaskedVarianceKernel = TRUE`.

continuous_info — Info for generating continuous covariates. Must be a list containing fields `lambda` and `mu`, which have length 3. The continuous covariates are generated so that `x <- sin(a*t + b) + c`, where

- `t <- seq(0, 2*pi, length.out = k)`
- `a <- mu[1] + lambda[1]*stats::runif(1)`
- `b <- mu[2] + lambda[2]*stats::runif(1)`
- `c <- mu[3] + lambda[3]*stats::runif(1)`

Value

A list `out`, where

- `out$data` is a data frame containing the actual data and
- `out$components` contains more points for smoother visualizations of the generating process.
- `out$onsets` contains the real disease effect times
- `out$p_signal` proportion of variance explained by signal

Examples

```
# Generate Gaussian data
dat <- simulate_data(N = 4, t_data = c(6,12,24,36,48), snr = 3)

# Generate negative binomially distributed count data
dat <- simulate_data(N = 6, t_data = seq(2, 10, by = 2), noise_type = "NB", phi = 2)
```

`simulate_kernels` *Compute all kernel matrices when simulating data*

Description

Compute all kernel matrices when simulating data

Usage

```
simulate_kernels(X, types, lengthscales, X_affected, useBinKernel,
  useMaskedVarianceKernel, steepness, vm_params)
```

Arguments

`X`	covariates
`types`	vector of covariate types, so that • 1 = ID • 2 = age • 3 = diseaseAge • 4 = other continuous covariate • 5 = a categorical covariate that interacts with age • 6 = a categorical covariate that acts as an offset
`lengthscales`	vector of lengthscales
`X_affected`	which individuals are affected by the disease
`useBinKernel`	whether or not binary (mask) kernel should be used for categorical covariates
`useMaskedVarianceKernel`	should the masked variance kernel be used for drawing the disease component
`steepness`	steepness of the input warping function
`vm_params`	parameters of the variance mask function

Value

a 3D array

sim_check_covariates *Input check for the covariates-related arguments of* simulate_data

Description

Input check for the covariates-related arguments of simulate_data

Usage

```
sim_check_covariates(covariates, relevances, names, n_cat)
```

Arguments

covariates	argument to simulate_data
relevances	argument to simulate_data
names	argument to simulate_data
n_cat	the n_categs argument to simulate_data

Value

the covariate names

sim_data_to_observed *Real generated disease ages to observed ones*

Description

Real generated disease ages to observed ones

Usage

```
sim_data_to_observed(dat, t_observed)
```

Arguments

dat	data frame
t_observed	Determines how the disease onset is observed. See documentation of simulate_data.

Value

a new data frame and observed onsets

sim_generate_names *Generate names for covariates*

Description

Generate names for covariates

Usage

```
sim_generate_names(covariates)
```

Arguments

covariates	vector of covariate types

Value

covariate names

sim_parse_t_obs *Parse the t_observed argument of* simulate_data

Description

Parse the t_observed argument of simulate_data

Usage

```
sim_parse_t_obs(t_observed)
```

Arguments

t_observed	a string

Value

a list with a name and number

split_data *Split data into training and test data according to given row indices*

Description

Split data into training and test data according to given row indices

Usage

```
split_data(data, i_test, sort_ids = TRUE)
```

Arguments

data	a data frame
i_test	test data row indices
sort_ids	should the test indices be sorted into increasing order

Value

a `list(train, test)`

split_data_by_id *Split data into training and test data according to given individuals*

Description

Split data into training and test data according to given individuals

Usage

```
split_data_by_id(data, test_ids, id_variable = "id")
```

Arguments

data	a data frame
test_ids	test data individual identifiers
id_variable	name of id variable

Value

a `list(train, test)`

split_data_by_timepoint *Split data into training and test data according to time point indices*

Description

Split data into training and test data according to time point indices

Usage

```
split_data_by_timepoint(data, test_idx, id_variable = "id",
  time_variable = "age")
```

Arguments

data	a data frame
test_idx	indices of test time points
id_variable	name of id variable
time_variable	name of time variable

Value

a list(train, test)

split_data_random *Split data into training and test data randomly*

Description

Split data into training and test data randomly

Usage

```
split_data_random(data, p_test = 0.1, n_test = NULL)
```

Arguments

data	a data frame
p_test	desired proportion of test data
n_test	desired number of test data points (if NULL, p_test is used to compute this)

Value

a list(train, test)

split_data_random_each

Split data into training and test data by selecting randomly k points from each individual

Description

Split data into training and test data by selecting randomly k points from each individual

Usage

```
split_data_random_each(data, n_test = 1, id_variable = "id",
  time_variable = "age")
```

Arguments

data	a data frame
n_test	desired number of test data points per individual
id_variable	name of id variable
time_variable	name of time variable

Value

a list(train, test)

standardize_inputs

Standardize continuous input variables in X

Description

Standardize continuous input variables in X

Usage

```
standardize_inputs(X, D)
```

Arguments

X	the design matrix
D	the covariate types, a vector of length 6

Value

updated X and info about scaling

stan_input_X_and_D *Predictor covariates and types to Stan input*

Description

Reorders covariates and takes only those that are needed

Usage

```
stan_input_X_and_D(data, varInfo, types, formula, verbose)
```

Arguments

data	a data frame containing the covariates
varInfo	original variable type info
types	types of the covariates
formula	model formula
verbose	can this print some info?

Value

X and needed types and updated varInfo

validate_prior *Validate prior by sampling the signal and noise from it*

Description

Validate prior by sampling the signal and noise from it

Usage

```
validate_prior(model, chains = 4, iter = 1000, parallel = FALSE)
```

Arguments

model	An object of class lgpmodel.
chains	how many chains are used to sample from the prior
iter	for how many iterations are the chains run
parallel	should the chains be run in parallel?

Value

An object of class lgpfit and random samples of both ‘f‘ and ‘y‘.

varsel *Covariate selection*

Description

Covariate selection

Usage

```
varsel(object, threshold = 0.95, verbose = TRUE)
```

Arguments

object	An object of class lgpfit.
threshold	A threshold for proportion of explained variance
verbose	should this print some output

Value

the selected covariates

warp_input *Warp inputs*

Description

Warp inputs

Usage

```
warp_input(t, a, b, c)
```

Arguments

t	a vector
a	steepness of the rise
b	location of the effective time window
c	maximum range

Value

a vector of warped inputs w(t)

Index