Package 'MagmaClustR'

June 6, 2022

Title Clustering and Prediction using Multi-Task Gaussian Processes with Common Mean

Version 1.0.0

Description An implementation for the multi-task Gaussian processes with common mean framework. Two main algorithms, called 'Magma' and 'MagmaClust', are available to perform predictions for supervised learning problems, in particular for time series or any functional/continuous data applications. The corresponding articles has been respectively proposed by Arthur Leroy, Pierre Latouche, Benjamin Guedj and Servane Gey (2022) <doi:10.1007/s10994-022-06172-1>, and Arthur Leroy, Pierre Latouche, Benjamin Guedj and Servane Gey (2020) <arXiv:2011.07866>. Theses approaches leverage the learning of cluster-specific mean processes, which are common across similar tasks, to provide enhanced prediction performances (even far from data) at a linear computational cost (in the number of tasks). 'MagmaClust' is a generalisation of 'Magma' where the tasks are simultaneously clustered into groups, each being associated to a specific mean process. User-oriented functions in the package are decomposed into training, prediction and plotting functions. Some basic features (classic kernels, training, prediction) of standard Gaussian processes are also implemented.

License MIT + file LICENSE

URL https://github.com/ArthurLeroy/MagmaClustR, https://arthurleroy.github.io/MagmaClustR/

BugReports https://github.com/ArthurLeroy/MagmaClustR/issues

Imports broom, dplyr, ggplot2, magrittr, methods, mvtnorm, optimr, Rcpp, rlang, stats, tibble, tidyr, tidyselect

Suggests gganimate, gifski, knitr, plotly, png, rmarkdown, testthat (>= 3.0.0), transformr

LinkingTo Rcpp

Encoding UTF-8

RoxygenNote 7.1.2

NeedsCompilation yes

Author Arthur Leroy [aut, cre] (<https://orcid.org/0000-0003-0806-8934>),
Pierre Pathé [ctb],
Pierre Latouche [aut]

Maintainer Arthur Leroy <arthur.leroy.pro@gmail.com>

Repository CRAN

Date/Publication 2022-06-06 09:00:10 UTC

R topics documented:

chol_inv_jitter . . . 3
data_allocate_cluster . . . 4
dmnorm . . . 4
draw . . . 5
elbo_clust_multi_GP . . . 5
elbo_clust_multi_GP_common_hp_i . . . 6
elbo_GP_mod_common_hp_k . . . 7
elbo_monitoring_VEM . . . 8
e_step . . . 9
gr_clust_multi_GP . . . 9
gr_clust_multi_GP_common_hp_i . . . 10
gr_GP . . . 11
gr_GP_mod . . . 12
gr_GP_mod_common_hp . . . 12
gr_GP_mod_common_hp_k . . . 13
gr_sum_logL_GP_clust . . . 14
hp . . . 15
hyperposterior . . . 16
hyperposterior_clust . . . 18
ini_kmeans . . . 20
ini_mixture . . . 20
kern_to_cov . . . 21
kern_to_inv . . . 22
lin_kernel . . . 23
list_kern_to_cov . . . 24
list_kern_to_inv . . . 25
logL_GP . . . 25
logL_GP_mod . . . 26
logL_GP_mod_common_hp . . . 27
logL_monitoring . . . 28
MagmaClustR . . . 29
m_step . . . 30
perio_kernel . . . 31
plot_db . . . 32
plot_gif . . . 32
plot_gp . . . 34
plot_magmaclust . . . 36
pred_gif . . . 38

pred_gp 40
pred_magma 42
pred_magmaclust 44
proba_max_cluster 46
rq_kernel 47
sample_gp 47
select_nb_cluster 49
se_kernel 50
simu_db 51
simu_indiv_se 53
sum_logL_GP_clust 53
train_gp 54
train_gp_clust 56
train_magma 58
train_magmaclust 61
update_mixture 64
ve_step 64
vm_step 65

Index **67**

`chol_inv_jitter` *Inverse a matrix using an adaptive jitter term*

Description

Inverse a matrix from its Choleski decomposition. If (nearly-)singular, increase the order of magnitude of the jitter term added to the diagonal until the matrix becomes non-singular.

Usage

```
chol_inv_jitter(mat, pen_diag)
```

Arguments

`mat`	A matrix, possibly singular.
`pen_diag`	A number, a jitter term to add on the diagonal.

Value

A matrix, inverse of `mat` plus an adaptive jitter term added on the diagonal.

Examples

```
TRUE
```

data_allocate_cluster *Allocate training data into the most probable cluster*

Description

Allocate training data into the most probable cluster

Usage

```
data_allocate_cluster(trained_model)
```

Arguments

trained_model	A list, containing the information coming from a MagmaClust model, previously trained using the train_magmaclust function.

Value

The original dataset used to train the MagmaClust model, with additional 'Cluster' and associated 'Proba' columns, indicating the most probable cluster for each individual/task at the end of the training procedure.

Examples

```
TRUE
```

dmnorm *Compute the Multivariate Gaussian likelihood*

Description

Modification of the function dmvnorm() from the package mvtnorm, providing an implementation of the Multivariate Gaussian likelihood. This version uses inverse of the covariance function as argument instead of the traditional covariance.

Usage

```
dmnorm(x, mu, inv_Sigma, log = FALSE)
```

Arguments

x	A vector, containing values the likelihood is evaluated on.
mu	A vector or matrix, specifying the mean parameter.
inv_Sigma	A matrix, specifying the inverse of covariance parameter.
log	A logical value, indicating whether we return the log-likelihood.

Value

A number, corresponding to the Multivariate Gaussian log-likelihood.

Examples

```
TRUE
```

`draw` *Draw a number*

Description

Draw uniformly a number within a specified interval

Usage

```
draw(int)
```

Arguments

`int`	An interval of values we want to draw uniformly in.

Value

A 2-decimals-rounded random number

Examples

```
TRUE
```

`elbo_clust_multi_GP` *Evidence Lower Bound for a mixture of GPs*

Description

Evidence Lower Bound for a mixture of GPs

Usage

```
elbo_clust_multi_GP(hp, db, hyperpost, kern, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing the values we want to compute the elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
hyperpost	List of parameters for the K mean GPs.
kern	A kernel function used to compute the covariance matrix at corresponding timestamps.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

The value of the penalised Gaussian elbo for a mixture of GPs

Examples

```
TRUE
```

---

elbo_clust_multi_GP_common_hp_i

Penalised elbo for multiple individual GPs with common HPs

Description

Penalised elbo for multiple individual GPs with common HPs

Usage

```
elbo_clust_multi_GP_common_hp_i(hp, db, hyperpost, kern, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing values we want to compute elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
hyperpost	List of parameters for the K mean Gaussian processes.
kern	A kernel function used to compute the covariance matrix at corresponding timestamps.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

The value of the penalised Gaussian elbo for the sum of the M individual GPs with common HPs.

Examples

```
TRUE
```

`elbo_GP_mod_common_hp_k` *Penalised elbo for multiple mean GPs with common HPs*

Description

Penalised elbo for multiple mean GPs with common HPs

Usage

```
elbo_GP_mod_common_hp_k(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

`hp`	A tibble, data frame or named vector containing hyper-parameters.
`db`	A tibble containing values we want to compute elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
`mean`	A list of the K mean GPs at union of observed timestamps.
`kern`	A kernel function used to compute the covariance matrix at corresponding timestamps.
`post_cov`	A List of the K posterior covariance of the mean GP (mu_k). Used to compute correction term (cor_term).
`pen_diag`	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

The value of the penalised Gaussian elbo for the sum of the k mean GPs with common HPs.

Examples

```
TRUE
```

`elbo_monitoring_VEM` *Evidence Lower Bound maximised in MagmaClust*

Description

Evidence Lower Bound maximised in MagmaClust

Usage

```
elbo_monitoring_VEM(hp_k, hp_i, db, kern_i, kern_k, hyperpost, m_k, pen_diag)
```

Arguments

`hp_k`	A tibble, data frame or named vector of hyper-parameters for each clusters.
`hp_i`	A tibble, data frame or named vector of hyper-parameters for each individuals.
`db`	A tibble containing values we want to compute elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
`kern_i`	Kernel used to compute the covariance matrix of individuals GPs at corresponding inputs.
`kern_k`	Kernel used to compute the covariance matrix of the mean GPs at corresponding inputs.
`hyperpost`	A list of parameters for the variational distributions of the K mean GPs.
`m_k`	Prior value of the mean parameter of the mean GPs (mu_k). Length = 1 or nrow(db).
`pen_diag`	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

Value of the elbo that is maximised during the VEM algorithm used for training in MagmaClust.

Examples

```
TRUE
```

e_step *E-Step of the EM algorithm*

Description

Expectation step of the EM algorithm to compute the parameters of the hyper-posterior Gaussian distribution of the mean process in Magma.

Usage

```
e_step(db, m_0, kern_0, kern_i, hp_0, hp_i, pen_diag)
```

Arguments

db	A tibble or data frame. Columns required: ID, Input, Output. Additional columns for covariates can be specified.
m_0	A vector, corresponding to the prior mean of the mean GP.
kern_0	A kernel function, associated with the mean GP.
kern_i	A kernel function, associated with the individual GPs.
hp_0	A named vector, tibble or data frame of hyper-parameters associated with kern_0.
hp_i	A tibble or data frame of hyper-parameters associated with kern_i.
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A named list, containing the elements mean, a tibble containing the Input and associated Output of the hyper-posterior's mean parameter, and cov, the hyper-posterior's covariance matrix.

Examples

```
TRUE
```

gr_clust_multi_GP *Gradient of the elbo for a mixture of GPs*

Description

Gradient of the elbo for a mixture of GPs

Usage

```
gr_clust_multi_GP(hp, db, hyperpost, kern, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing the values we want to compute the elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
hyperpost	List of parameters for the K mean Gaussian processes.
kern	A kernel function.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

The gradient of the penalised Gaussian elbo for a mixture of GPs

Examples

```
TRUE
```

---

gr_clust_multi_GP_common_hp_i

Gradient of the penalised elbo for multiple individual GPs with common HPs

---

Description

Gradient of the penalised elbo for multiple individual GPs with common HPs

Usage

```
gr_clust_multi_GP_common_hp_i(hp, db, hyperpost, kern, pen_diag = NULL)
```

Arguments

hp	A tibble, data frame or name vector of hyper-parameters.
db	A tibble containing values we want to compute elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
hyperpost	List of parameters for the K mean Gaussian processes.
kern	A kernel function used to compute the covariance matrix at corresponding timestamps.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

The gradient of the penalised Gaussian elbo for the sum of the M individual GPs with common HPs.

Examples

```
TRUE
```

---

gr_GP *Gradient of the logLikelihood of a Gaussian Process*

Description

Gradient of the logLikelihood of a Gaussian Process

Usage

```
gr_GP(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing the values we want to compute the logL on. Required columns: Input, Output. Additional covariate columns are allowed.
mean	A vector, specifying the mean of the GP at the reference inputs.
kern	A kernel function.
post_cov	(optional) A matrix, corresponding to covariance parameter of the hyper-posterior. Used to compute the hyper-prior distribution of a new individual in Magma.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A named vector, corresponding to the value of the hyper-parameters gradients for the Gaussian log-Likelihood (where the covariance can be the sum of the individual and the hyper-posterior's mean process covariances).

Examples

```
TRUE
```

gr_GP_mod *Gradient of the modified logLikelihood for GPs in Magma*

Description

Gradient of the modified logLikelihood for GPs in Magma

Usage

```
gr_GP_mod(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing the values we want to compute the logL on. Required columns: Input, Output. Additional covariate columns are allowed.
mean	A vector, specifying the mean of the GPs at the reference inputs.
kern	A kernel function.
post_cov	A matrix, covariance parameter of the hyper-posterior. Used to compute the correction term.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A named vector, corresponding to the value of the hyper-parameters gradients for the modified Gaussian log-Likelihood involved in Magma.

Examples

```
TRUE
```

gr_GP_mod_common_hp *Gradient of the modified logLikelihood with common HPs for GPs in Magma*

Description

Gradient of the modified logLikelihood with common HPs for GPs in Magma

Usage

```
gr_GP_mod_common_hp(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

hp	A tibble or data frame containing hyper-parameters for all individuals.
db	A tibble containing the values we want to compute the logL on. Required columns: ID, Input, Output. Additional covariate columns are allowed.
mean	A vector, specifying the mean of the GPs at the reference inputs.
kern	A kernel function.
post_cov	A matrix, covariance parameter of the hyper-posterior. Used to compute the correction term.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A named vector, corresponding to the value of the hyper-parameters' gradients for the modified Gaussian log-Likelihood involved in Magma with the 'common HP' setting.

Examples

```
TRUE
```

---

gr_GP_mod_common_hp_k *Gradient of the penalised elbo for multiple mean GPs with common HPs*

Description

Gradient of the penalised elbo for multiple mean GPs with common HPs

Usage

```
gr_GP_mod_common_hp_k(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing the values we want to compute the elbo on. Required columns: Input, Output. Additional covariate columns are allowed.
mean	A list of the k means of the GPs at union of observed timestamps.
kern	A kernel function
post_cov	A list of the k posterior covariance of the mean GP (mu_k). Used to compute correction term (cor_term)
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

The gradient of the penalised Gaussian elbo for the sum of the k mean GPs with common HPs.

Examples

```
TRUE
```

`gr_sum_logL_GP_clust` *Gradient of the mixture of Gaussian likelihoods*

Description

Compute the gradient of a sum of Gaussian log-likelihoods, weighted by their mixture probabilities.

Usage

```
gr_sum_logL_GP_clust(hp, db, mixture, mean, kern, post_cov, pen_diag)
```

Arguments

`hp`	A tibble, data frame or named vector of hyper-parameters.
`db`	A tibble containing data we want to evaluate the logL on. Required columns: Input, Output. Additional covariate columns are allowed.
`mixture`	A tibble or data frame, indicating the mixture probabilities of each cluster for the new individual/task.
`mean`	A list of hyper-posterior mean parameters for all clusters.
`kern`	A kernel function.
`post_cov`	A list of hyper-posterior covariance parameters for all clusters.
`pen_diag`	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A named vector, corresponding to the value of the hyper-parameters' gradients for the mixture of Gaussian log-likelihoods involved in the prediction step of MagmaClust.

Examples

```
TRUE
```

hp *Generate random hyper-parameters*

Description

Generate a set of random hyper-parameters, specific to the chosen type of kernel, under the format that is used in Magma.

Usage

```
hp(
  kern = "SE",
  list_ID = NULL,
  list_hp = NULL,
  noise = FALSE,
  common_hp = FALSE
)
```

Arguments

kern	A function, or a character string indicating the chosen type of kernel among: • "SE": the Squared Exponential kernel, • "LIN": the Linear kernel, • "PERIO": the Periodic kernel, • "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not). In case of a custom kernel function, the argument list_hp has to be provided as well, for designing a tibble with the correct names of hyper-parameters.
list_ID	A vector, associating an ID value with each individual for whom hyper-parameters are generated. If NULL (default) only one set of hyper-parameters is return without the ID column.
list_hp	A vector of characters, providing the name of each hyper-parameter, in case where kern is a custom kernel function.
noise	A logical value, indicating whether a 'noise' hyper-parameter should be included.
common_hp	A logical value, indicating whether the set of hyper-parameters is assumed to be common to all indiviuals.

Value

A tibble, providing a set of random hyper-parameters associated with the kernel specified through the argument `kern`.

Examples

```
TRUE
```

`hyperposterior` *Compute the hyper-posterior distribution in Magma*

Description

Compute the parameters of the hyper-posterior Gaussian distribution of the mean process in Magma (similarly to the expectation step of the EM algorithm used for learning). This hyper-posterior distribution, evaluated on a grid of inputs provided through the `grid_inputs` argument, is a key component for making prediction in Magma, and is required in the function `pred_magma`.

Usage

```
hyperposterior(
  data,
  hp_0,
  hp_i,
  kern_0,
  kern_i,
  prior_mean = NULL,
  grid_inputs = NULL,
  pen_diag = 1e-10
)
```

Arguments

`data`	A tibble or data frame. Required columns: 'Input', 'Output'. Additional columns for covariates can be specified. The 'Input' column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The 'Output' column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference 'Input'.
`hp_0`	A named vector, tibble or data frame of hyper-parameters associated with `kern_0`.
`hp_i`	A tibble or data frame of hyper-parameters associated with `kern_i`.
`kern_0`	A kernel function, associated with the mean GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`kern_i` A kernel function, associated with the individual GPs. ("SE", "PERIO" and "RQ" are aso available here)

`prior_mean` Hyper-prior mean parameter of the mean GP. This argument, can be specified under various formats, such as:

- NULL (default). The hyper-prior mean would be set to 0 everywhere.
- A number. The hyper-prior mean would be a constant function.
- A vector of the same length as all the distinct Input values in the `data` argument. This vector would be considered as the evaluation of the hyper-prior mean function at the training Inputs.
- A function. This function is defined as the hyper-prior mean.
- A tibble or data frame. Required columns: Input, Output. The Input values should include at least the same values as in the `data` argument.

`grid_inputs` A vector, indicating the grid of additional reference inputs on which the mean process' hyper-posterior should be evaluated.

`pen_diag` A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A list gathering the parameters of the mean processes' hyper-posterior distributions, namely:

- mean: A tibble, the hyper-posterior mean parameter evaluated at each training `Input`.
- cov: A matrix, the covariance parameter for the hyper-posterior distribution of the mean process.
- pred: A tibble, the predicted mean and variance at `Input` for the mean process' hyper-posterior distribution under a format that allows the direct visualisation as a GP prediction.

Examples

```
TRUE
```

`hyperposterior_clust` *Compute the hyper-posterior distribution for each cluster in MagmaClust*

Description

Recompute the E-step of the VEM algorithm in MagmaClust for a new set of reference `Input`. Once training is completed, it can be necessary to evaluate the hyper-posterior distributions of the mean processes at specific locations, for which we want to make predictions. This process is directly implemented in the `pred_magmaclust` function but for the user might want to use `hyperpost_clust` for a tailored control 'by hand' of the prediction procedure.

Usage

```
hyperposterior_clust(
  data,
  mixture,
  hp_k,
  hp_i,
  kern_k,
  kern_i,
  prior_mean_k = NULL,
  grid_inputs = NULL,
  pen_diag = 1e-10
)
```

Arguments

`data`	A tibble or data frame. Required columns: `ID`, `Input` , `Output`. Additional columns for covariates can be specified. The `ID` column contains the unique names/codes used to identify each individual/task (or batch of data). The `Input` column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The `Output` column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference `Input`.
`mixture`	A tibble or data frame, indicating the mixture probabilities of each cluster for each individual. Required column: `ID`.
`hp_k`	A tibble or data frame of hyper-parameters associated with `kern_k`.
`hp_i`	A tibble or data frame of hyper-parameters associated with `kern_i`.
`kern_k`	A kernel function, associated with the mean GPs. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list: • "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),

- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`kern_i` A kernel function, associated with the individual GPs. ("SE", "LIN", PERIO" and "RQ" are also available here)

`prior_mean_k` The set of hyper-prior mean parameters (m_k) for the K mean GPs, one value for each cluster. cluster. This argument can be specified under various formats, such as:

- NULL (default). All hyper-prior means would be set to 0 everywhere.
- A numerical vector of the same length as the number of clusters. Each number is associated with one cluster, and considered to be the hyper-prior mean parameter of the cluster (i.e. a constant function at all `Input`).
- A list of functions. Each function is associated with one cluster. These functions are all evaluated at all `Input` values, to provide specific hyper-prior mean vectors for each cluster.

`grid_inputs` A vector, indicating the grid of additional reference inputs on which the mean process' hyper-posterior should be evaluated.

`pen_diag` A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A list containing the parameters of the mean processes' hyper-posterior distribution, namely:

- mean: A list of tibbles containing, for each cluster, the hyper-posterior mean parameters evaluated at each `Input`.
- cov: A list of matrices containing, for each cluster, the hyper-posterior covariance parameter of the mean process.
- mixture: A tibble, indicating the mixture probabilities in each cluster for each individual.

Examples

```
TRUE
```

ini_kmeans *Run a k-means algoithm to initialise clusters' allocation*

Description

Run a k-means algoithm to initialise clusters' allocation

Usage

```
ini_kmeans(data, k, nstart = 50, summary = FALSE)
```

Arguments

data	A tibble containing common Input and associated Output values to cluster.
k	A number of clusters assumed for running the kmeans algorithm.
nstart	A number, indicating how many re-starts of kmeans are set.
summary	A boolean, indicating whether we want an outcome summary

Value

A tibble containing the initial clustering obtained through kmeans.

Examples

```
TRUE
```

ini_mixture *Mixture initialisation with kmeans*

Description

Provide an initial kmeans allocation of the individuals/tasks in a dataset into a definite number of clusters, and return the associated mixture probabilities.

Usage

```
ini_mixture(data, k, name_clust = NULL, nstart = 50)
```

Arguments

data	A tibble or data frame. Required columns: ID, Input , Output.
k	A number, indicating the number of clusters.
name_clust	A vector of characters. Each element should correspond to the name of one cluster.
nstart	A number of restart used in the underlying kmeans algorithm

Value

A tibble indicating for each `ID` in which cluster it belongs after a kmeans initialisation.

Examples

```
TRUE
```

`kern_to_cov` *Create covariance matrix from a kernel*

Description

`kern_to_cov()` creates a covariance matrix between input values (that could be either scalars or vectors) evaluated within a kernel function, which is characterised by specified hyper-parameters. This matrix is a finite-dimensional evaluation of the infinite-dimensional covariance structure of a GP, defined thanks to this kernel.

Usage

```
kern_to_cov(input, kern = "SE", hp, deriv = NULL, input_2 = NULL)
```

Arguments

`input`	A vector, matrix, data frame or tibble containing all inputs for one individual. If a vector, the elements are used as reference, otherwise , one column should be named 'Input' to indicate that it represents the reference (e.g. 'Input' would contain the timestamps in time-series applications). The other columns are considered as being covariates. If no column is named 'Input', the first one is used by default.
`kern`	A kernel function. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list: • "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel), • "LIN": the Linear kernel, • "PERIO": the Periodic kernel, • "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).
`hp`	A list, data frame or tibble containing the hyper-parameters used in the kernel. The name of the elements (or columns) should correspond exactly to those used in the kernel definition. If `hp` contains an element or a column 'Noise', its value will be added on the diagonal of the covariance matrix.

deriv — A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the covariance matrix.

input_2 — (optional) A vector, matrix, data frame or tibble under the same format as input. This argument should be used only when the kernel needs to be evaluated between two different sets of inputs, typically resulting in a non-square matrix.

Value

A covariance matrix, where elements are evaluations of the associated kernel for each pair of reference inputs.

Examples

```
TRUE
```

---

kern_to_inv *Create inverse of a covariance matrix from a kernel*

Description

kern_to_inv() creates the inverse of a covariance matrix between input values (that could be either scalars or vectors) evaluated within a kernel function, which is characterised by specified hyper-parameters. This matrix is a finite-dimensional evaluation of the infinite-dimensional covariance structure of a GP, defined thanks to this kernel.

Usage

```
kern_to_inv(input, kern, hp, pen_diag = 1e-10, deriv = NULL)
```

Arguments

input — A vector, matrix, data frame or tibble containing all inputs for one individual. If a vector, the elements are used as reference, otherwise ,one column should be named 'Input' to indicate that it represents the reference (e.g. 'Input' would contain the timestamps in time-series applications). The other columns are considered as being covariates. If no column is named 'Input', the first one is used by default.

kern — A kernel function. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,

- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`hp` A list, data frame or tibble containing the hyper-parameters used in the kernel. The name of the elements (or columns) should correspond exactly to those used in the kernel definition.

`pen_diag` A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

`deriv` A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the inverse covariance matrix.

Value

The inverse of a covariance matrix, which elements are evaluations of the associated kernel for each pair of reference inputs.

Examples

```
TRUE
```

---

lin_kernel *Linear Kernel*

Description

Linear Kernel

Usage

```
lin_kernel(x, y, hp, deriv = NULL, vectorized = FALSE)
```

Arguments

`x` A vector (or matrix if vectorized = T) of inputs.

`y` A vector (or matrix if vectorized = T) of inputs.

`hp` A tibble, data frame or named vector, containing the kernel's hyperparameters. Required columns: 'lin_slope' and 'lin_offset'.

`deriv` A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the evaluation of the kernel.

vectorized — A logical value, indicating whether the function provides a vectorized version for speeded-up calculations. If TRUE, the x and y arguments should be the vector or matrix containing all inputs for which the kernel is evaluated on all pairs of elements. If FALSE, the x and y arguments are simply two inputs.

Value

A scalar, corresponding to the evaluation of the kernel.

Examples

```
TRUE
```

---

list_kern_to_cov *Compute a covariance matrix for multiple individuals*

Description

Compute the covariance matrices associated with all individuals in the database, taking into account their specific inputs and hyper-parameters.

Usage

```
list_kern_to_cov(data, kern, hp, deriv = NULL)
```

Arguments

data	A tibble or data frame of input data. Required column: 'ID'. Suggested column: 'Input' (for indicating the reference input).
kern	A kernel function.
hp	A tibble or data frame, containing the hyper-parameters associated with each individual.
deriv	A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the list of covariance matrices.

Value

A named list containing all of the inverse covariance matrices.

Examples

```
TRUE
```

list_kern_to_inv *Compute an inverse covariance matrix for multiple individuals*

Description

Compute the inverse covariance matrices associated with all individuals in the database, taking into account their specific inputs and hyper-parameters.

Usage

```
list_kern_to_inv(db, kern, hp, pen_diag, deriv = NULL)
```

Arguments

db	A tibble or data frame of input data. Required column: 'ID'. Suggested column: 'Input' (for indicating the reference input).
kern	A kernel function.
hp	A tibble or data frame, containing the hyper-parameters associated with each individual.
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.
deriv	A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the list of covariance matrices.

Value

A named list containing all of the inverse covariance matrices.

Examples

```
TRUE
```

logL_GP *Log-Likelihood function of a Gaussian Process*

Description

Log-Likelihood function of a Gaussian Process

Usage

```
logL_GP(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector containing hyper-parameters.
db	A tibble containing the values we want to compute the logL on. Required columns: Input, Output. Additional covariate columns are allowed.
mean	A vector, specifying the mean of the GP at the reference inputs.
kern	A kernel function.
post_cov	(optional) A matrix, corresponding to covariance parameter of the hyper-posterior. Used to compute the hyper-prior distribution of a new individual in Magma.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A number, corresponding to the value of Gaussian log-Likelihood (where the covariance can be the sum of the individual and the hyper-posterior's mean process covariances).

Examples

```
TRUE
```

---

logL_GP_mod *Modified log-Likelihood function for GPs*

Description

Log-Likelihood function involved in Magma during the maximisation step of the training. The log-Likelihood is defined as a simple Gaussian likelihood added with correction trace term.

Usage

```
logL_GP_mod(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

hp	A tibble, data frame or named vector of hyper-parameters.
db	A tibble containing values we want to compute logL on. Required columns: Input, Output. Additional covariate columns are allowed.
mean	A vector, specifying the mean of the GP at the reference inputs.
kern	A kernel function.
post_cov	A matrix, covariance parameter of the hyper-posterior. Used to compute the correction term.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A number, corresponding to the value of the modified Gaussian log-Likelihood defined in Magma.

Examples

```
TRUE
```

`logL_GP_mod_common_hp` *Modified log-Likelihood function with common HPs for GPs*

Description

Log-Likelihood function involved in Magma during the maximisation step of the training, in the particular case where the hyper-parameters are shared by all individuals. The log-Likelihood is defined as a sum over all individuals of Gaussian likelihoods added with correction trace terms.

Usage

```
logL_GP_mod_common_hp(hp, db, mean, kern, post_cov, pen_diag)
```

Arguments

`hp`	A tibble, data frame of hyper-parameters.
`db`	A tibble containing the values we want to compute the logL on. Required columns: ID, Input, Output. Additional covariate columns are allowed.
`mean`	A vector, specifying the mean of the GP at the reference inputs.
`kern`	A kernel function.
`post_cov`	A matrix, covariance parameter of the hyper-posterior. Used to compute the correction term.
`pen_diag`	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A number, corresponding to the value of the modified Gaussian log-Likelihood with common hyper-parameters defined in Magma.

Examples

```
TRUE
```

logL_monitoring *Log-Likelihood for monitoring the EM algorithm in Magma*

Description

Log-Likelihood for monitoring the EM algorithm in Magma

Usage

```
logL_monitoring(
  hp_0,
  hp_i,
  db,
  m_0,
  kern_0,
  kern_i,
  post_mean,
  post_cov,
  pen_diag
)
```

Arguments

hp_0	A named vector, tibble or data frame, containing the hyper-parameters associated with the mean GP.
hp_i	A tibble or data frame, containing the hyper-parameters with the individual GPs.
db	A tibble or data frame. Columns required: ID, Input, Output. Additional columns for covariates can be specified.
m_0	A vector, corresponding to the prior mean of the mean GP.
kern_0	A kernel function, associated with the mean GP.
kern_i	A kernel function, associated with the individual GPs.
post_mean	A tibble, coming out of the E step, containing the Input and associated Output of the hyper-posterior mean parameter.
post_cov	A matrix, coming out of the E step, being the hyper-posterior covariance parameter.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A number, expectation of joint log-likelihood of the model. This quantity is supposed to increase at each step of the EM algorithm, and thus used for monitoring the procedure.

Examples

```
TRUE
```

MagmaClustR *MagmaClustR : Clustering and Prediction using Multi-Task Gaussian Processes*

Description

The **MagmaClustR** package implements two main algorithms, called *Magma* and *MagmaClust*, using a multi-task GPs model to perform predictions for supervised learning problems. Theses approaches leverage the learning of cluster-specific mean processes, which are common across similar tasks, to provide enhanced prediction performances (even far from data) at a linear computational cost (in the number of tasks). *MagmaClust* is a generalisation of *Magma* where the tasks are simultaneously clustered into groups, each being associated to a specific mean process. User-oriented functions in the package are decomposed into training, prediction and plotting functions. Some basic features of standard GPs are also implemented.

Details

For a quick introduction to **MagmaClustR**, please refer to the README at https://github.com/ArthurLeroy/MagmaClustR

Author(s)

Arthur Leroy, Pierre Pathe and Pierre Latouche
Maintainer: Arthur Leroy - <arthur.leroy.pro@gmail.com>

References

Arthur Leroy, Pierre Latouche, Benjamin Guedj, and Servane Gey.
MAGMA: Inference and Prediction with Multi-Task Gaussian Processes. *Machine Learning*, 2022, https://link.springer.com/article/10.1007/s10994-022-06172-1

Arthur Leroy, Pierre Latouche, Benjamin Guedj, and Servane Gey.
Cluster-Specific Predictions with Multi-Task Gaussian Processes. *PREPRINT*, Nov. 2020, https://arxiv.org/abs/2011.07866

Examples

Simulate a dataset, train and predict with Magma :

```
set.seed(42)
data_magma <- simu_db(M = 11, N = 10, K = 1)
magma_train <- data_magma %>% subset(ID %in% 1:10)
magma_test <- data_magma %>% subset(ID == 11) %>% head(5)

magma_model <- train_magma(data = magma_train)
magma_pred <- pred_magma(data = magma_test, trained_model = magma_model, grid_inputs =
seq(0, 10, 0.01))
```

Simulate a dataset, train and predict with MagmaClust
:

```
set.seed(42)
data_magmaclust <- simu_db(M = 4, N = 10, K = 3)
list_ID = unique(data_magmaclust$ID)
magmaclust_train <- data_magmaclust %>% subset(ID %in% list_ID[1:11])
magmaclust_test <- data_magmaclust %>% subset(ID == list_ID[12]) %>% head(5)

magmaclust_model <- train_magmaclust(data = magmaclust_train)
magmaclust_pred <- pred_magmaclust(data = magmaclust_test,
trained_model = magmaclust_model, grid_inputs = seq(0, 10, 0.01))
```

`m_step` *M-Step of the EM algorithm*

Description

Maximisation step of the EM algorithm to compute hyper-parameters of all the kernels involved in Magma.

Usage

```
m_step(
  db,
  m_0,
  kern_0,
  kern_i,
  old_hp_0,
  old_hp_i,
  post_mean,
  post_cov,
  common_hp,
  pen_diag
)
```

Arguments

`db`	A tibble or data frame. Columns required: ID, Input, Output. Additional columns for covariates can be specified.
`m_0`	A vector, corresponding to the prior mean of the mean GP.
`kern_0`	A kernel function, associated with the mean GP.
`kern_i`	A kernel function, associated with the individual GPs.
`old_hp_0`	A named vector, tibble or data frame, containing the hyper-parameters from the previous M-step (or initialisation) associated with the mean GP.

old_hp_i	A tibble or data frame, containing the hyper-parameters from the previous M-step (or initialisation) associated with the individual GPs.
post_mean	A tibble, coming out of the E step, containing the Input and associated Output of the hyper-posterior mean parameter.
post_cov	A matrix, coming out of the E step, being the hyper-posterior covariance parameter.
common_hp	A logical value, indicating whether the set of hyper-parameters is assumed to be common to all indiviuals.
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A named list, containing the elements hp_0, a tibble containing the hyper-parameters associated with the mean GP, hp_i, a tibble containing the hyper-parameters associated with the individual GPs.

Examples

```
TRUE
```

---

perio_kernel *Periodic Kernel*

Description

Periodic Kernel

Usage

```
perio_kernel(x, y, hp, deriv = NULL, vectorized = FALSE)
```

Arguments

x	A vector (or matrix if vectorized = T) of inputs.
y	A vector (or matrix if vectorized = T) of inputs.
hp	A tibble, data frame or named vector, containing the kernel's hyperparameters. Required columns: 'perio_variance', 'perio_lengthscale', and 'period'.
deriv	A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the evaluation of the kernel.
vectorized	A logical value, indicating whether the function provides a vectorized version for speeded-up calculations. If TRUE, the x and y arguments should be the vector or matrix containing all inputs for which the kernel is evaluated on all pairs of elements. If FALSE, the x and y arguments are simply two inputs.

Value

A scalar, corresponding to the evaluation of the kernel.

Examples

```
TRUE
```

`plot_db` *Plot smoothed curves of raw data*

Description

Display raw data under the Magma format as smoothed curves.

Usage

```
plot_db(data, cluster = FALSE, legend = FALSE)
```

Arguments

`data`	A data frame or tibble with format : ID, Input, Output.
`cluster`	A boolean indicating whether data should be coloured by cluster. Requires a column named 'Cluster'.
`legend`	A boolean indicating whether the legend should be displayed.

Value

Graph of smoothed curves of raw data.

Examples

```
TRUE
```

`plot_gif` *Create a GIF of Magma or GP predictions*

Description

Create a GIF animation displaying how Magma or classic GP predictions evolve and improve when the number of data points increase.

Usage

```
plot_gif(
  pred_gp,
  x_input = NULL,
  data = NULL,
  data_train = NULL,
  prior_mean = NULL,
  y_grid = NULL,
  heatmap = FALSE,
  prob_CI = 0.95,
  size_data = 3,
  size_data_train = 1,
  alpha_data_train = 0.5,
  export_gif = FALSE,
  path = "gif_gp.gif",
  ...
)
```

Arguments

pred_gp	A tibble, typically coming from the pred_gif function. Required columns: 'Input', 'Mean', 'Var' and 'Index'.
x_input	A vector of character strings, indicating which input should be displayed. If NULL(default) the 'Input' column is used for the x-axis. If providing a 2-dimensional vector, the corresponding columns are used for the x-axis and y-axis.
data	(Optional) A tibble or data frame. Required columns: 'Input', 'Output'. Additional columns for covariates can be specified. The 'Input' column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The 'Output' column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference 'Input'.
data_train	(Optional) A tibble or data frame, containing the training data of the Magma model. The data set should have the same format as the data argument with an additional column 'ID' for identifying the different individuals/tasks. If provided, those data are displayed as backward colourful points (each colour corresponding to one individual/task).
prior_mean	(Optional) A tibble or a data frame, containing the 'Input' and associated 'Output' prior mean parameter of the GP prediction.
y_grid	A vector, indicating the grid of values on the y-axis for which probabilities should be computed for heatmaps of 1-dimensional predictions. If NULL (default), a vector of length 50 is defined, ranging between the min and max 'Output' values contained in pred_gp.

heatmap	A logical value indicating whether the GP prediction should be represented as a heatmap of probabilities for 1-dimensional inputs. If FALSE (default), the mean curve and associated 95% CI are displayed.
prob_CI	A number between 0 and 1 (default is 0.95), indicating the level of the Credible Interval associated with the posterior mean curve.
size_data	A number, controlling the size of the data points.
size_data_train	A number, controlling the size of the data_train points.
alpha_data_train	A number, between 0 and 1, controlling transparency of the data_train points.
export_gif	A logical value indicating whether the animation should be exported as a .gif file.
path	A character string defining the path where the GIF file should be exported.
...	Any additional parameters that can be passed to the function transition_states from the gganimate package.

Value

Visualisation of a Magma or GP prediction (optional: display data points, training data points and the prior mean function), where data points are added sequentially for visualising changes in prediction as information increases.

Examples

```
TRUE
```

---

plot_gp *Plot Magma or GP predictions*

Description

Display Magma or classic GP predictions. According to the dimension of the inputs, the graph may be a mean curve + Credible Interval or a heatmap of probabilities.

Usage

```
plot_gp(
  pred_gp,
  x_input = NULL,
  data = NULL,
  data_train = NULL,
  prior_mean = NULL,
  y_grid = NULL,
  heatmap = FALSE,
  prob_CI = 0.95,
```

```
  size_data = 3,
  size_data_train = 1,
  alpha_data_train = 0.5
)
```

Arguments

pred_gp	A tibble or data frame, typically coming from pred_magma or pred_gp functions. Required columns: 'Input', 'Mean', 'Var'. Additional covariate columns may be present in case of multi-dimensional inputs.
x_input	A vector of character strings, indicating which input should be displayed. If NULL (default) the 'Input' column is used for the x-axis. If providing a 2-dimensional vector, the corresponding columns are used for the x-axis and y-axis.
data	(Optional) A tibble or data frame. Required columns: 'Input', 'Output'. Additional columns for covariates can be specified. This argument corresponds to the raw data on which the prediction has been performed.
data_train	(Optional) A tibble or data frame, containing the training data of the Magma model. The data set should have the same format as the data argument with an additional required column 'ID' for identifying the different individuals/tasks. If provided, those data are displayed as backward colourful points (each colour corresponding to one individual/task).
prior_mean	(Optional) A tibble or a data frame, containing the 'Input' and associated 'Output' prior mean parameter of the GP prediction.
y_grid	A vector, indicating the grid of values on the y-axis for which probabilities should be computed for heatmaps of 1-dimensional predictions. If NULL (default), a vector of length 50 is defined, ranging between the min and max 'Output' values contained in pred_gp.
heatmap	A logical value indicating whether the GP prediction should be represented as a heatmap of probabilities for 1-dimensional inputs. If FALSE (default), the mean curve and associated Credible Interval are displayed.
prob_CI	A number between 0 and 1 (default is 0.95), indicating the level of the Credible Interval associated with the posterior mean curve. If this this argument is set to 1, the Credible Interval is not displayed.
size_data	A number, controlling the size of the data points.
size_data_train	A number, controlling the size of the data_train points.
alpha_data_train	A number, between 0 and 1, controlling transparency of the data_train points.

Value

Visualisation of a Magma or GP prediction (optional: display data points, training data points and the prior mean function). For 1-D inputs, the prediction is represented as a mean curve and its associated 95% Credible Interval, or as a heatmap of probabilities if heatmap = TRUE. For 2-D

inputs, the prediction is represented as a heatmap, where each couple of inputs on the x-axis and y-axis are associated with a gradient of colours for the posterior mean values, whereas the uncertainty is indicated by the transparency (the narrower is the Credible Interval, the more opaque is the associated colour, and vice versa)

Examples

```
TRUE
```

`plot_magmaclust` *Plot MagmaClust predictions*

Description

Display MagmaClust predictions. According to the dimension of the inputs, the graph may be a mean curve (dim inputs = 1) or a heatmap (dim inputs = 2) of probabilities. Moreover, MagmaClust can provide credible intervals only by visualising cluster-specific predictions (e.g. for the most probable cluster). When visualising the full mixture-of-GPs prediction, which can be multimodal, the user should choose between the simple mean function or the full heatmap of probabilities (more informative but slower).

Usage

```
plot_magmaclust(
  pred_clust,
  cluster = "all",
  x_input = NULL,
  data = NULL,
  data_train = NULL,
  col_clust = FALSE,
  prior_mean = NULL,
  y_grid = NULL,
  heatmap = FALSE,
  prob_CI = 0.95,
  size_data = 3,
  size_data_train = 1,
  alpha_data_train = 0.5
)
```

Arguments

`pred_clust`	A list of predictions, typically coming from `pred_magmaclust`. Required elements: `pred`, `mixture`, `mixture_pred`.
`cluster`	A character string, indicating which cluster to plot from. If 'all' (default) the mixture of GPs prediction is displayed as a mean curve (1-D inputs) or a mean heatmap (2-D inputs). Alternatively, if the name of one cluster is provided, the classic mean curve + credible interval is displayed (1-D inputs), or a heatmap

with colour gradient for the mean and transparency gradient for the Credible Interval (2-D inputs).

`x_input` A vector of character strings, indicating which input should be displayed. If NULL (default) the 'Input' column is used for the x-axis. If providing a 2-dimensional vector, the corresponding columns are used for the x-axis and y-axis.

`data` (Optional) A tibble or data frame. Required columns: `Input` , `Output`. Additional columns for covariates can be specified. This argument corresponds to the raw data on which the prediction has been performed.

`data_train` (Optional) A tibble or data frame, containing the training data of the MagmaClust model. The data set should have the same format as the `data` argument with an additional required column `ID` for identifying the different individuals/tasks. If provided, those data are displayed as backward colourful points (each colour corresponding to one individual or a cluster, see `col_clust` below).

`col_clust` A boolean indicating whether backward points are coloured according to the individuals or to their most probable cluster. If one wants to colour by clusters, a column `Cluster` shall be present in `data_train`. We advise to use `data_allocate_cluster` for automatically creating a well-formatted dataset from a trained MagmaClust model.

`prior_mean` (Optional) A list providing, for each cluster, a tibble containing prior mean parameters of the prediction. This argument typically comes as an outcome `hyperpost$mean`, available through the `train_magmaclust`, `pred_magmaclust` functions.

`y_grid` A vector, indicating the grid of values on the y-axis for which probabilities should be computed for heatmaps of 1-dimensional predictions. If NULL (default), a vector of length 50 is defined, ranging between the min and max 'Output' values contained in `pred`.

`heatmap` A logical value indicating whether the GP prediction should be represented as a heatmap of probabilities for 1-dimensional inputs. If FALSE (default), the mean curve (and associated Credible Interval if available) are displayed.

`prob_CI` A number between 0 and 1 (default is 0.95), indicating the level of the Credible Interval associated with the posterior mean curve. If this this argument is set to 1, the Credible Interval is not displayed.

`size_data` A number, controlling the size of the `data` points.

`size_data_train`
A number, controlling the size of the `data_train` points.

`alpha_data_train`
A number, between 0 and 1, controlling transparency of the `data_train` points.

Value

Visualisation of a MagmaClust prediction (optional: display data points, training data points and the prior mean functions). For 1-D inputs, the prediction is represented as a mean curve (and its associated 95% Credible Interval for cluster-specific predictions), or as a heatmap of probabilities if `heatmap = TRUE`. In the case of MagmaClust, the heatmap representation should be preferred

for clarity, although the default display remains mean curve for quicker execution. For 2-D inputs, the prediction is represented as a heatmap, where each couple of inputs on the x-axis and y-axis are associated with a gradient of colours for the posterior mean values, whereas the uncertainty is indicated by the transparency (the narrower is the Credible Interval, the more opaque is the associated colour, and vice versa). As for 1-D inputs, Credible Interval information is only available for cluster-specific predictions.

Examples

```
TRUE
```

`pred_gif` *Magma prediction for ploting GIFs*

Description

Generate a Magma or classic GP prediction under a format that is compatible with a further GIF visualisation of the results. For a Magma prediction, either the `trained_model` or `hyperpost` argument is required. Otherwise, a classic GP prediction is applied and the prior mean can be specified through the `mean` argument.

Usage

```
pred_gif(
  data,
  trained_model = NULL,
  hyperpost = NULL,
  mean = NULL,
  hp = NULL,
  kern = "SE",
  grid_inputs = NULL,
  pen_diag = 1e-10
)
```

Arguments

`data`	A tibble or data frame. Required columns: 'Input', 'Output'. Additional columns for covariates can be specified. The 'Input' column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The 'Output' column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference 'Input'.
`trained_model`	A list, containing the information coming from a Magma model, previously trained using the `train_magma` function.

`hyperpost` A list, containing the elements 'mean' and 'cov', the parameters of the hyper-posterior distribution of the mean process. Typically, this argument should from a previous learning using `train_magma`, or a previous prediction with `pred_magma`, with the argument `get_hyperpost` set to TRUE. The 'mean' element should be a data frame with two columns 'Input' and 'Output'. The 'cov' element should be a covariance matrix with colnames and rownames corresponding to the 'Input' in 'mean'. In all cases, the column 'Input' should contain all the values appearing both in the 'Input' column of `data` and in `grid_inputs`.

`mean` Mean parameter of the GP. This argument can be specified under various formats, such as:

- NULL (default). The mean would be set to 0 everywhere.
- A number. The mean would be a constant function.
- A function. This function is defined as the mean.
- A tibble or data frame. Required columns: Input, Output. The Input values should include at least the same values as in the `data` argument.

`hp` A named vector, tibble or data frame of hyper-parameters associated with `kern`. The columns/elements should be named according to the hyper-parameters that are used in `kern`. The function `train_gp` can be used to learn maximum-likelihood estimators of the hyper-parameters,

`kern` A kernel function, defining the covariance structure of the GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`grid_inputs` The grid of inputs (reference Input and covariates) values on which the GP should be evaluated. Ideally, this argument should be a tibble or a data frame, providing the same columns as `data`, except 'Output'. Nonetheless, in cases where `data` provides only one 'Input' column, the `grid_inputs` argument can be NULL (default) or a vector. This vector would be used as reference input for prediction and if NULL, a vector of length 500 is defined, ranging between the min and max Input values of `data`.

`pen_diag` A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A tibble, representing Magma or GP predictions as two column 'Mean' and 'Var', evaluated on the `grid_inputs`. The column 'Input' and additional covariates columns are associated to each

predicted values. An additional 'Index' column is created for the sake of GIF creation using the function plot_gif

Examples

```
TRUE
```

---

pred_gp *Gaussian Process prediction*

Description

Compute the posterior distribution of a simple GP, using the formalism of Magma. By providing observed data, the prior mean and covariance matrix (by defining a kernel and its associated hyper-parameters), the mean and covariance parameters of the posterior distribution are computed on the grid of inputs that has been specified. This predictive distribution can be evaluated on any arbitrary inputs since a GP is an infinite-dimensional object.

Usage

```
pred_gp(
  data,
  mean = NULL,
  hp = NULL,
  kern = "SE",
  grid_inputs = NULL,
  get_full_cov = FALSE,
  plot = TRUE,
  pen_diag = 1e-10
)
```

Arguments

data	A tibble or data frame. Required columns: 'Input', 'Output'. Additional columns for covariates can be specified. The 'Input' column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The 'Output' column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference 'Input'.
mean	Mean parameter of the GP. This argument can be specified under various formats, such as: • NULL (default). The mean would be set to 0 everywhere. • A number. The mean would be a constant function. • A function. This function is defined as the mean. • A tibble or data frame. Required columns: Input, Output. The Input values should include at least the same values as in the data argument.

hp — A named vector, tibble or data frame of hyper-parameters associated with kern. The columns/elements should be named according to the hyper-parameters that are used in kern. If NULL (default), the function train_gp is called with random initial values for learning maximum-likelihood estimators of the hyper-parameters associated with kern.

kern — A kernel function, defining the covariance structure of the GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

grid_inputs — The grid of inputs (reference Input and covariates) values on which the GP should be evaluated. Ideally, this argument should be a tibble or a data frame, providing the same columns as data, except 'Output'. Nonetheless, in cases where data provides only one 'Input' column, the grid_inputs argument can be NULL (default) or a vector. This vector would be used as reference input for prediction and if NULL, a vector of length 500 is defined, ranging between the min and max Input values of data.

get_full_cov — A logical value, indicating whether the full posterior covariance matrix should be returned.

plot — A logical value, indicating whether a plot of the results is automatically displayed.

pen_diag — A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A tibble, representing the GP predictions as two column 'Mean' and 'Var', evaluated on the grid_inputs. The column 'Input' and additional covariates columns are associated to each predicted values. If the get_full_cov argument is TRUE, the function returns a list, in which the tibble described above is defined as 'pred' and the full posterior covariance matrix is defined as 'cov'.

Examples

```
TRUE
```

pred_magma *Magma prediction*

Description

Compute the posterior predictive distribution in Magma. Providing data of any new individual/task, its trained hyper-parameters and a previously trained Magma model, the predictive distribution is evaluated on any arbitrary inputs that are specified through the 'grid_inputs' argument.

Usage

```
pred_magma(
  data,
  trained_model = NULL,
  hp = NULL,
  kern = "SE",
  grid_inputs = NULL,
  hyperpost = NULL,
  get_hyperpost = FALSE,
  get_full_cov = FALSE,
  plot = TRUE,
  pen_diag = 1e-10
)
```

Arguments

data	A tibble or data frame. Required columns: 'Input', 'Output'. Additional columns for covariates can be specified. The 'Input' column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The 'Output' column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference 'Input'.
trained_model	A list, containing the information coming from a Magma model, previously trained using the train_magma function.
hp	A named vector, tibble or data frame of hyper-parameters associated with kern. The columns/elements should be named according to the hyper-parameters that are used in kern. The function train_gp can be used to learn maximum-likelihood estimators of the hyper-parameters.
kern	A kernel function, defining the covariance structure of the GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list: • "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel), • "LIN": the Linear kernel,

- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`grid_inputs` The grid of inputs (reference Input and covariates) values on which the GP should be evaluated. Ideally, this argument should be a tibble or a data frame, providing the same columns as `data`, except 'Output'. Nonetheless, in cases where `data` provides only one 'Input' column, the `grid_inputs` argument can be NULL (default) or a vector. This vector would be used as reference input for prediction and if NULL, a vector of length 500 is defined, ranging between the min and max Input values of `data`.

`hyperpost` A list, containing the elements 'mean' and 'cov', the parameters of the hyper-posterior distribution of the mean process. Typically, this argument should come from a previous learning using `train_magma`, or a previous prediction with `pred_magma`, with the argument `get_hyperpost` set to TRUE. The 'mean' element should be a data frame with two columns 'Input' and 'Output'. The 'cov' element should be a covariance matrix with colnames and rownames corresponding to the 'Input' in 'mean'. In all cases, the column 'Input' should contain all the values appearing both in the 'Input' column of `data` and in `grid_inputs`.

`get_hyperpost` A logical value, indicating whether the hyper-posterior distribution of the mean process should be returned. This can be useful when planning to perform several predictions on the same grid of inputs, since recomputation of the hyper-posterior can be prohibitive for high dimensional grids.

`get_full_cov` A logical value, indicating whether the full posterior covariance matrix should be returned.

`plot` A logical value, indicating whether a plot of the results is automatically displayed.

`pen_diag` A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A tibble, representing Magma predictions as two column 'Mean' and 'Var', evaluated on the `grid_inputs`. The column 'Input' and additional covariates columns are associated to each predicted values. If the `get_full_cov` or `get_hyperpost` arguments are TRUE, the function returns a list, in which the tibble described above is defined as 'pred_gp' and the full posterior covariance matrix is defined as 'cov', and the hyper-posterior distribution of the mean process is defined as 'hyperpost'.

Examples

```
TRUE
```

`pred_magmaclust` *MagmaClust prediction*

Description

Compute the posterior predictive distribution in MagmaClust. Providing data from any new individual/task, its trained hyper-parameters and a previously trained MagmaClust model, the multi-task posterior distribution is evaluated on any arbitrary inputs that are specified through the 'grid_inputs' argument. Due to the nature of the model, the prediction is defined as a mixture of Gaussian distributions. Therefore the present function computes the parameters of the predictive distribution associated with each cluster, as well as the posterior mixture probabilities for this new individual/task.

Usage

```
pred_magmaclust(
  data,
  trained_model = NULL,
  mixture = NULL,
  hp = NULL,
  kern = "SE",
  grid_inputs = NULL,
  hyperpost = NULL,
  prop_mixture = NULL,
  get_hyperpost = FALSE,
  get_full_cov = FALSE,
  plot = TRUE,
  pen_diag = 1e-10
)
```

Arguments

`data`	A tibble or data frame. Required columns: `Input`, `Output`. Additional columns for covariates can be specified. The `Input` column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The `Output` column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference 'Input'.
`trained_model`	A list, containing the information coming from a MagmaClust model, previously trained using the `train_magmaclust` function. If `trained_model` is set to NULL, the `hyperpost` and `prop_mixture` arguments are mandatory to perform required re-computations for the prediction to succeed.
`mixture`	A tibble or data frame, indicating the mixture probabilities of each cluster for the new individual/task. If NULL, the `train_gp_clust` function is used to compute these posterior probabilities according to `data`.

`hp` A named vector, tibble or data frame of hyper-parameters associated with `kern`. The columns/elements should be named according to the hyper-parameters that are used in `kern`. The `train_gp_clust` function can be used to learn maximum-likelihood estimators of the hyper-parameters.

`kern` A kernel function, defining the covariance structure of the GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`grid_inputs` The grid of inputs (reference Input and covariates) values on which the GP should be evaluated. Ideally, this argument should be a tibble or a data frame, providing the same columns as `data`, except 'Output'. Nonetheless, in cases where `data` provides only one 'Input' column, the `grid_inputs` argument can be NULL (default) or a vector. This vector would be used as reference input for prediction and if NULL, a vector of length 500 is defined, ranging between the min and max Input values of `data`.

`hyperpost` A list, containing the elements `mean`, `cov` and `mixture` the parameters of the hyper-posterior distributions of the mean processes. Typically, this argument should come from a previous learning using `train_magmaclust`, or a previous prediction with `pred_magmaclust`, with the argument `get_hyperpost` set to TRUE.

`prop_mixture` A tibble or a named vector of the mixture proportions. Each name of column or element should refer to a cluster. The value associated with each cluster is a number between 0 and 1. If both `mixture` and `trained_model` are set to NULL, this argument allows to recompute mixture probabilities, thanks to the `hyperpost` argument and the `train_gp_clust` function.

`get_hyperpost` A logical value, indicating whether the hyper-posterior distributions of the mean processes should be returned. This can be useful when planning to perform several predictions on the same grid of inputs, since recomputation of the hyper-posterior can be prohibitive for high dimensional grids.

`get_full_cov` A logical value, indicating whether the full posterior covariance matrices should be returned.

`plot` A logical value, indicating whether a plot of the results is automatically displayed.

`pen_diag` A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A list of GP prediction results composed of:

- pred: As sub-list containing, for each cluster:
  - pred_gp: A tibble, representing the GP predictions as two column `Mean` and `Var`, evaluated on the `grid_inputs`. The column `Input` and additional covariates columns are associated with each predicted values.
  - proba: A number, the posterior probability associated with this cluster.
  - cov (if `get_full_cov` = TRUE): A matrix, the full posterior covariance matrix associated with this cluster.
- mixture: A tibble, indicating the mixture probabilities of each cluster for the predicted individual/task.
- hyperpost (if `get_hyperpost` = TRUE): A list, containing the hyper-posterior distributions information useful for visualisation purposes.

Examples

```
TRUE
```

---

proba_max_cluster *Indicates the most probable cluster*

Description

Indicates the most probable cluster

Usage

```
proba_max_cluster(mixture)
```

Arguments

`mixture`	A tibble or data frame containing mixture probabilities.

Value

A tibble, retaining only the most probable cluster. The column `Cluster` indicates the the cluster's name whereas `Proba` refers to its associated probability. If `ID` is initially a column of `mixture` (optional), the function returns the most probable cluster for all the different `ID` values.

Examples

```
TRUE
```

rq_kernel *Rational Quadratic Kernel*

Description

Rational Quadratic Kernel

Usage

```
rq_kernel(x, y, hp, deriv = NULL, vectorized = FALSE)
```

Arguments

x	A vector (or matrix if vectorized = T) of inputs.
y	A vector (or matrix if vectorized = T) of inputs.
hp	A tibble, data frame or named vector, containing the kernel's hyperparameters. Required columns: 'rq_variance', 'rq_lengthscale', and 'rq_scale'.
deriv	A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the evaluation of the kernel.
vectorized	A logical value, indicating whether the function provides a vectorized version for speeded-up calculations. If TRUE, the x and y arguments should be the vector or matrix containing all inputs for which the kernel is evaluated on all pairs of elements. If FALSE, the x and y arguments are simply two inputs.

Value

A scalar, corresponding to the evaluation of the kernel.

Examples

```
TRUE
```

sample_gp *Display Realisation From Posterior GP*

Description

A realisation of a posterior GP distribution is drawn and displayed. According to the dimension of the inputs, the graph may be a curve or a heatmap.

Usage

```
sample_gp(
  pred_gp,
  x_input = NULL,
  data = NULL,
  data_train = NULL,
  prior_mean = NULL,
  size_data = 3,
  size_data_train = 1,
  alpha_data_train = 0.5
)
```

Arguments

pred_gp	A tibble or data frame, typically coming from pred_magma or pred_gp functions. Required columns: 'Input', 'Mean', 'Var'. Additional covariate columns may be present in case of multi-dimensional inputs.
x_input	A vector of character strings, indicating which input should be displayed. If NULL(default) the 'Input' column is used for the x-axis. If providing a 2-dimensional vector, the corresponding columns are used for the x-axis and y-axis.
data	(Optional) A tibble or data frame, containing the data used in the GP prediction.
data_train	(Optional) A tibble or data frame, containing the training data of the Magma model. The data set should have the same format as the data argument with an additional column 'ID' for identifying the different individuals/tasks. If provided, those data are displayed as backward colourful points (each colour corresponding to one individual/task).
prior_mean	(Optional) A tibble or a data frame, containing the 'Input' and associated 'Output' prior mean parameter of the GP prediction.
size_data	A number, controlling the size of the data points.
size_data_train	A number, controlling the size of the data_train points.
alpha_data_train	A number, between 0 and 1, controlling transparency of the data_train points.

Value

Draw and visualise from a posterior distribution from Magma or GP prediction (optional: display data points, training data points and the prior mean function).

Examples

```
TRUE
```

`select_nb_cluster` *Select the optimal number of clusters*

Description

In MagmaClust, as for any clustering method, the number K of clusters has to be provided as an hypothesis of the model. This function implements a model selection procedure, by maximising a variational BIC criterion, computed for different values of K. A heuristic for a fast approximation of the procedure is proposed as well, although the corresponding models would not be properly trained.

Usage

```
select_nb_cluster(
  data,
  fast_approx = TRUE,
  grid_nb_cluster = 1:10,
  ini_hp_k = NULL,
  ini_hp_i = NULL,
  kern_k = "SE",
  kern_i = "SE",
  plot = TRUE,
  ...
)
```

Arguments

`data`	A tibble or data frame. Columns required: `ID`, `Input` , `Output`. Additional columns for covariates can be specified. The `ID` column contains the unique names/codes used to identify each individual/task (or batch of data). The `Input` column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The `Output` column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference `Input`.
`fast_approx`	A boolean, indicating whether a fast approximation should be used for selecting the number of clusters. If TRUE, each Magma or MagmaClust model will perform only one E-step of the training, using the same fixed values for the hyper-parameters (`ini_hp_k` and `ini_hp_i`, or random values if not provided) in all models. The resulting models should not be considered as trained, but this approach provides an convenient heuristic to avoid a cumbersome model selection procedure.
`grid_nb_cluster`	A vector of integer, corresponding to grid of values that will be tested for the number of clusters.

ini_hp_k	A tibble or data frame of hyper-parameters associated with kern_k.
ini_hp_i	A tibble or data frame of hyper-parameters associated with kern_i.
kern_k	A kernel function associated to the mean processes.
kern_i	A kernel function associated to the individuals/tasks.
plot	A boolean indicating whether the plot of V-BIC values for all numbers of clusters should displayed.
...	Any additional argument that could be passed to train_magmaclust.

Value

A list, containing the results of model selection procedure for selecting the optimal number of clusters thanks to a V-BIC criterion maximisation. The elements of the list are:

- best_k: An integer, indicating the resulting optimal number of clusters
- seq_vbic: A vector, corresponding to the sequence of the V-BIC values associated with the models trained for each provided cluster's number in grid_nb_cluster.
- trained_models: A list, named by associated number of clusters, of Magma or MagmaClust models that have been trained (or approximated if fast_approx = T) during the model selection procedure.

Examples

```
TRUE
```

---

se_kernel *Squared Exponential Kernel*

Description

Squared Exponential Kernel

Usage

```
se_kernel(x, y, hp, deriv = NULL, vectorized = FALSE)
```

Arguments

x	A vector (or matrix if vectorized = T) of inputs.
y	A vector (or matrix if vectorized = T) of inputs.
hp	A tibble, data frame or named vector, containing the kernel's hyperparameters. Required columns: 'se_variance', 'se_lengthscale'.
deriv	A character, indicating according to which hyper-parameter the derivative should be computed. If NULL (default), the function simply returns the evaluation of the kernel.
vectorized	A logical value, indicating whether the function provides a vectorized version for speeded-up calculations. If TRUE, the x and y arguments should be the vector or matrix containing all inputs for which the kernel is evaluated on all pairs of elements. If FALSE, the x and y arguments are simply two inputs.

Value

A scalar, corresponding to the evaluation of the kernel.

Examples

```
TRUE
```

`simu_db` *Simulate a dataset tailored for MagmaClustR*

Description

Simulate a complete training dataset, which may be representative of various applications. Several flexible arguments allow adjustment of the number of individuals, of observed inputs, and the values of many parameters controlling the data generation.

Usage

```
simu_db(
  M = 10,
  N = 10,
  K = 1,
  covariate = FALSE,
  grid = seq(0, 10, 0.05),
  common_input = TRUE,
  common_hp = TRUE,
  add_hp = FALSE,
  add_clust = FALSE,
  int_mu_v = c(0, 2),
  int_mu_l = c(0, 2),
  int_i_v = c(0, 2),
  int_i_l = c(0, 2),
  int_i_sigma = c(0, 1),
  m0_slope = c(-5, 5),
  m0_intercept = c(-10, 10),
  int_covariate = c(-5, 5)
)
```

Arguments

`M`	An integer. The number of individual per cluster.
`N`	An integer. The number of observations per individual.
`K`	An integer. The number of underlying clusters.
`covariate`	A logical value indicating whether the dataset should include an additional input covariate named 'Covariate'.
`grid`	A vector of numbers defining a grid of observations (i.e. the reference inputs).

common_input	A logical value indicating whether the reference inputs are common to all individual.
common_hp	A logical value indicating whether the hyper-parameters are common to all individual. If TRUE and K>1, the hyper-parameters remain different between the clusters.
add_hp	A logical value indicating whether the values of hyper-parameters should be added as columns in the dataset.
add_clust	A logical value indicating whether the name of the clusters should be added as a column in the dataset.
int_mu_v	A vector of 2 numbers, defining an interval of admissible values for the variance hyper-parameter of the mean process' kernel.
int_mu_l	A vector of 2 numbers, defining an interval of admissible values for the length-scale hyper-parameter of the mean process' kernel.
int_i_v	A vector of 2 numbers, defining an interval of admissible values for the variance hyper-parameter of the individual process' kernel.
int_i_l	A vector of 2 numbers, defining an interval of admissible values for the length-scale hyper-parameter of the individual process' kernel.
int_i_sigma	A vector of 2 numbers, defining an interval of admissible values for the noise hyper-parameter.
m0_slope	A vector of 2 numbers, defining an interval of admissible values for the slope of m0.
m0_intercept	A vector of 2 numbers, defining an interval of admissible values for the intercept of m0.
int_covariate	A vector of 2 numbers, defining an interval of admissible values for the covariate inputs.

Value

A full dataset of simulated training data.

Examples

```
## Generate a dataset with 3 clusters of 4 individuals, observed at 10 inputs
data = simu_db(M = 4, N = 10, K = 3)

## Generate a 2-D dataset with an additional input 'Covariate'
data = simu_db(covariate = TRUE)

## Generate a dataset where input locations are different among individuals
data = simu_db(common_input = FALSE)

## Generate a dataset with an additional column indicating the true clusters
data = simu_db(K = 3, covariate = TRUE)
```

simu_indiv_se *Simulate a batch a data*

Description

Simulate a batch a output data, corresponding to one individual, coming from a GP with a the Squared Exponential kernel as covariance structure, and specified hyper-parameters and input.

Usage

```
simu_indiv_se(ID, input, covariate, mean, v, l, sigma)
```

Arguments

ID	An identification code, whether numeric or character.
input	A vector of numbers. The input variable that is used as 'reference' for input and outputs.
covariate	A vector of numbers. An additional input variable, observed along with each reference input.
mean	A vector of numbers. Prior mean values of the GP.
v	A number. The variance hyper-parameter of the SE kernel.
l	A number. The lengthscale hyper-parameter of the SE kernel.
sigma	A number. The noise hyper-parameter.

Value

A tibble containing a batch of output data along with input and additional information for a simulated individual.

Examples

```
TRUE
```

sum_logL_GP_clust *Compute a mixture of Gaussian log-likelihoods*

Description

During the prediction step of MagmaClust, an EM algorithm is used to compute the maximum likelihood estimator of the hyper-parameters along with mixture probabilities for the new individual/task. This function implements the quantity that is maximised (i.e. a sum of Gaussian log-likelihoods, weighted by their mixture probabilities). It can also be used to monitor the EM algorithm when providing the 'prop_mixture' argument, for proper penalisation of the full log-likelihood.

Usage

```
sum_logL_GP_clust(
  hp,
  db,
  mixture,
  mean,
  kern,
  post_cov,
  prop_mixture = NULL,
  pen_diag
)
```

Arguments

hp	A tibble, data frame or named vector of hyper-parameters.
db	A tibble containing data we want to evaluate the logL on. Required columns: Input, Output. Additional covariate columns are allowed.
mixture	A tibble or data frame, indicating the mixture probabilities of each cluster for the new individual/task.
mean	A list of hyper-posterior mean parameters for all clusters.
kern	A kernel function.
post_cov	A list of hyper-posterior covariance parameters for all clusters.
prop_mixture	A tibble or a named vector. Each name of column or element should refer to a cluster. The value associated with each cluster is a number between 0 and 1, corresponding to the mixture proportions.
pen_diag	A jitter term that is added to the covariance matrix to avoid numerical issues when inverting, in cases of nearly singular matrices.

Value

A number, expectation of mixture of Gaussian log-likelihoods in the prediction step of MagmaClust. This quantity is supposed to increase at each step of the EM algorithm, and can be used for monitoring the procedure.

Examples

```
TRUE
```

---

train_gp *Learning hyper-parameters of a Gaussian Process*

Description

Learning hyper-parameters of any new individual/task in Magma is required in the prediction procedure. This function can also be used to learn hyper-parameters of a simple GP (just let the hyperpost argument set to NULL, and use prior_mean instead). When using within Magma, by providing data for the new individual/task, the hyper-posterior mean and covariance parameters, and initialisation values for the hyper-parameters, the function computes maximum likelihood estimates of the hyper-parameters.

Usage

```
train_gp(
  data,
  prior_mean = NULL,
  ini_hp = NULL,
  kern = "SE",
  hyperpost = NULL,
  pen_diag = 1e-10
)
```

Arguments

data	A tibble or data frame. Required columns: Input, Output. Additional columns for covariates can be specified. The Input column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The Output column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference Input.
prior_mean	Mean parameter of the GP. This argument can be specified under various formats, such as: • NULL (default). The hyper-posterior mean would be set to 0 everywhere. • A number. The hyper-posterior mean would be a constant function. • A vector of the same length as all the distinct Input values in the data argument. This vector would be considered as the evaluation of the hyper-posterior mean function at the training Inputs. • A function. This function is defined as the hyper-posterior mean. • A tibble or data frame. Required columns: Input, Output. The Input values should include at least the same values as in the data argument.
ini_hp	A named vector, tibble or data frame of hyper-parameters associated with the kern of the new individual/task. The columns should be named according to the hyper-parameters that are used in kern. In cases where the model includes a noise term, ini_hp should contain an additional 'noise' column. If NULL (default), random values are used as initialisation.
kern	A kernel function, defining the covariance structure of the GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the² elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`hyperpost` A list, containing the elements 'mean' and 'cov', the parameters of the hyper-posterior distribution of the mean process. Typically, this argument should come from a previous learning using `train_magma`, or from the `hyperposterior` function. If `hyperpost` is provided, the likelihood that is maximised is the one involved during Magma's prediction step, and the `prior_mean` argument is ignored. For classic GP training, leave `hyperpost` to NULL.

`pen_diag` A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A tibble, containing the trained hyper-parameters for the kernel of the new individual/task.

Examples

```
TRUE
```

`train_gp_clust` *Prediction in MagmaClust: learning new HPs and mixture probabilities*

Description

Learning hyper-parameters and mixture probabilities of any new individual/task is required in `MagmaClust` in the prediction procedure. By providing data for the new individual/task, the hyper-posterior mean and covariance parameters, the mixture proportions, and initialisation values for the hyper-parameters, `train_gp_clust` uses an EM algorithm to compute maximum likelihood estimates of the hyper-parameters and hyper-posterior mixture probabilities of the new individual/task.

Usage

```
train_gp_clust(
  data,
  prop_mixture = NULL,
  ini_hp = NULL,
```

```
  kern = "SE",
  hyperpost = NULL,
  pen_diag = 1e-10,
  n_iter_max = 25,
  cv_threshold = 0.001
)
```

Arguments

data	A tibble or data frame. Required columns: `Input`, `Output`. Additional columns for covariates can be specified. The `Input` column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The `Output` column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference `Input`.
prop_mixture	A tibble or a named vector. Each name of column or element should refer to a cluster. The value associated with each cluster is a number between 0 and 1, corresponding to the mixture proportions.
ini_hp	A tibble or data frame of hyper-parameters associated with `kern`, the individual process kernel.
kern	A kernel function, defining the covariance structure of the GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list: • "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel), • "LIN": the Linear kernel, • "PERIO": the Periodic kernel, • "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the² elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).
hyperpost	A list, containing the elements `mean`, `cov` and `mixture` the parameters of the hyper-posterior distributions of the mean processes. Typically, this argument should come from a previous learning using `train_magmaclust`, or a previous prediction with `pred_magmaclust`, with the argument `get_hyperpost` set to TRUE.
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.
n_iter_max	A number, indicating the maximum number of iterations of the EM algorithm to proceed while not reaching convergence.
cv_threshold	A number, indicating the threshold of the likelihood gain under which the EM algorithm will stop.

Value

A list, containing the results of the EM algorithm used during the prediction step of MagmaClust. The elements of the list are:

- hp: A tibble of optimal hyper-parameters for the new individual's GP.
- mixture: A tibble of mixture probabilities for the new individual.

Examples

```
TRUE
```

`train_magma` *Training Magma with an EM algorithm*

Description

The hyper-parameters and the hyper-posterior distribution involved in Magma can be learned thanks to an EM algorithm implemented in `train_magma`. By providing a dataset, the model hypotheses (hyper-prior mean parameter and covariance kernels) and initialisation values for the hyper-parameters, the function computes maximum likelihood estimates of the HPs as well as the mean and covariance parameters of the Gaussian hyper-posterior distribution of the mean process.

Usage

```
train_magma(
  data,
  prior_mean = NULL,
  ini_hp_0 = NULL,
  ini_hp_i = NULL,
  kern_0 = "SE",
  kern_i = "SE",
  common_hp = TRUE,
  grid_inputs = NULL,
  pen_diag = 1e-10,
  n_iter_max = 25,
  cv_threshold = 0.001,
  fast_approx = FALSE
)
```

Arguments

`data`	A tibble or data frame. Required columns: `ID`, `Input` , `Output`. Additional columns for covariates can be specified. The `ID` column contains the unique names/codes used to identify each individual/task (or batch of data). The `Input` column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The `Output` column specifies the observed

values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference `Input`.

`prior_mean` Hyper-prior mean parameter (m_0) of the mean GP. This argument can be specified under various formats, such as:

- NULL (default). The hyper-prior mean would be set to 0 everywhere.
- A number. The hyper-prior mean would be a constant function.
- A vector of the same length as all the distinct Input values in the `data` argument. This vector would be considered as the evaluation of the hyper-prior mean function at the training Inputs.
- A function. This function is defined as the hyper_prior mean.
- A tibble or data frame. Required columns: Input, Output. The Input values should include at least the same values as in the `data` argument.

`ini_hp_0` A named vector, tibble or data frame of hyper-parameters associated with `kern_0`, the mean process' kernel. The columns/elements should be named according to the hyper-parameters that are used in `kern_0`. If NULL (default), random values are used as initialisation.

`ini_hp_i` A tibble or data frame of hyper-parameters associated with `kern_i`, the individual processes' kernel. Required column : `ID`. The `ID` column contains the unique names/codes used to identify each individual/task. The other columns should be named according to the hyper-parameters that are used in `kern_i`. Compared to `ini_hp_0` should contain an additional 'noise' column to initialise the noise hyper-parameter of the model. If NULL (default), random values are used as initialisation.

`kern_0` A kernel function, associated with the mean GP. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`kern_i` A kernel function, associated with the individual GPs. ("SE", "PERIO" and "RQ" are also available here).

`common_hp` A logical value, indicating whether the set of hyper-parameters is assumed to be common to all individuals.

`grid_inputs` A vector, indicating the grid of additional reference inputs on which the mean process' hyper-posterior should be evaluated.

pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.
n_iter_max	A number, indicating the maximum number of iterations of the EM algorithm to proceed while not reaching convergence.
cv_threshold	A number, indicating the threshold of the likelihood gain under which the EM algorithm will stop. The convergence condition is defined as the difference of likelihoods between two consecutive steps, divided by the absolute value of the last one ($(LL_n - LL_n - 1)/\|LL_n\|$).
fast_approx	A boolean, indicating whether the EM algorithm should stop after only one iteration of the E-step. This advanced feature is mainly used to provide a faster approximation of the model selection procedure, by preventing any optimisation over the hyper-parameters.

Details

The user can specify custom kernel functions for the argument `kern_0` and `kern_i`. The hyper-parameters used in the kernel should have explicit names, and be contained within the `hp` argument. `hp` should typically be defined as a named vector or a data frame. Although it is not mandatory for the `train_magma` function to run, gradients can be provided within kernel function definition. See for example `se_kernel` to create a custom kernel function displaying an adequate format to be used in Magma.

Value

A list, gathering the results of the EM algorithm used for training in Magma. The elements of the list are:

- hp_0: A tibble of the trained hyper-parameters for the mean process' kernel.
- hp_i: A tibble of all the trained hyper-parameters for the individual processes' kernels.
- hyperpost: A sub-list gathering the parameters of the mean processes' hyper-posterior distributions, namely:
  - mean: A tibble, the hyper-posterior mean parameter (`Output`) evaluated at each training reference `Input`.
  - cov: A matrix, the covariance parameter for the hyper-posterior distribution of the mean process.
  - pred: A tibble, the predicted mean and variance at `Input` for the mean process' hyper-posterior distribution under a format that allows the direct visualisation as a GP prediction.
- ini_args: A list containing the initial function arguments and values for the hyper-prior mean, the hyper-parameters. In particular, if those arguments were set to NULL, `ini_args` allows us to retrieve the (randomly chosen) initialisations used during training.
- seq_loglikelihood: A vector, containing the sequence of log-likelihood values associated with each iteration.
- converged: A logical value indicated whether the EM algorithm converged or not.
- training_time: Total running time of the complete training.

Examples

```
TRUE
```

`train_magmaclust` *Training MagmaClust with a Variational EM algorithm*

Description

The hyper-parameters and the hyper-posterior distributions involved in MagmaClust can be learned thanks to a VEM algorithm implemented in `train_magmaclust`. By providing a dataset, the model hypotheses (hyper-prior mean parameters, covariance kernels and number of clusters) and initialisation values for the hyper-parameters, the function computes maximum likelihood estimates of the HPs as well as the mean and covariance parameters of the Gaussian hyper-posterior distributions of the mean processes.

Usage

```
train_magmaclust(
  data,
  nb_cluster = NULL,
  prior_mean_k = NULL,
  ini_hp_k = NULL,
  ini_hp_i = NULL,
  kern_k = "SE",
  kern_i = "SE",
  ini_mixture = NULL,
  common_hp_k = TRUE,
  common_hp_i = TRUE,
  grid_inputs = NULL,
  pen_diag = 1e-10,
  n_iter_max = 25,
  cv_threshold = 0.001,
  fast_approx = FALSE
)
```

Arguments

`data`	A tibble or data frame. Columns required: `ID`, `Input` , `Output`. Additional columns for covariates can be specified. The `ID` column contains the unique names/codes used to identify each individual/task (or batch of data). The `Input` column should define the variable that is used as reference for the observations (e.g. time for longitudinal data). The `Output` column specifies the observed values (the response variable). The data frame can also provide as many covariates as desired, with no constraints on the column names. These covariates are additional inputs (explanatory variables) of the models that are also observed at each reference `Input`.

`nb_cluster` A number, indicating the number of clusters of individuals/tasks that are assumed to exist among the dataset.

`prior_mean_k` The set of hyper-prior mean parameters (m_k) for the K mean GPs, one value for each cluster. cluster. This argument can be specified under various formats, such as:

- NULL (default). All hyper-prior means would be set to 0 everywhere.
- A numerical vector of the same length as the number of clusters. Each number is associated with one cluster, and considered to be the hyper-prior mean parameter of the cluster (i.e. a constant function at all `Input`).
- A list of functions. Each function is associated with one cluster. These functions are all evaluated at all `Input` values, to provide specific hyper-prior mean vectors for each cluster.

`ini_hp_k` A tibble or data frame of hyper-parameters associated with `kern_k`, the mean process' kernel. Required column : `ID`. The `ID` column contains the unique names/codes used to identify each cluster. The other columns should be named according to the hyper-parameters that are used in `kern_k`.

`ini_hp_i` A tibble or data frame of hyper-parameters associated with `kern_i`, the individual processes' kernel. Required column : `ID`. The `ID` column contains the unique names/codes used to identify each individual/task. The other columns should be named according to the hyper-parameters that are used in `kern_i`.

`kern_k` A kernel function, associated with the mean GPs. Several popular kernels (see The Kernel Cookbook) are already implemented and can be selected within the following list:

- "SE": (default value) the Squared Exponential Kernel (also called Radial Basis Function or Gaussian kernel),
- "LIN": the Linear kernel,
- "PERIO": the Periodic kernel,
- "RQ": the Rational Quadratic kernel. Compound kernels can be created as sums or products of the above kernels. For combining kernels, simply provide a formula as a character string where elements are separated by whitespaces (e.g. "SE + PERIO"). As the elements are treated sequentially from the left to the right, the product operator '*' shall always be used before the '+' operators (e.g. 'SE * LIN + RQ' is valid whereas 'RQ + SE * LIN' is not).

`kern_i` A kernel function, associated with the individual GPs. (See details above in `kern_k`).

`ini_mixture` Initial values of the probability to belong to each cluster for each individual (`ini_mixture` can be used for a k-means initialisation. Used by default if NULL).

`common_hp_k` A boolean indicating whether hyper-parameters are common among the mean GPs.

`common_hp_i` A boolean indicating whether hyper-parameters are common among the individual GPs.

`grid_inputs` A vector, indicating the grid of additional reference inputs on which the mean processes' hyper-posteriors should be evaluated.

pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.
n_iter_max	A number, indicating the maximum number of iterations of the VEM algorithm to proceed while not reaching convergence.
cv_threshold	A number, indicating the threshold of the likelihood gain under which the VEM algorithm will stop. The convergence condition is defined as the difference of elbo between two consecutive steps, divided by the absolute value of the last one ($(ELBO_n - ELBO_{n-1})/\|ELBO_n\|$).
fast_approx	A boolean, indicating whether the VEM algorithm should stop after only one iteration of the VE-step. This advanced feature is mainly used to provide a faster approximation of the model selection procedure, by preventing any optimisation over the hyper-parameters.

Details

The user can specify custom kernel functions for the argument kern_k and kern_i. The hyper-parameters used in the kernel should have explicit names, and be contained within the hp argument. hp should typically be defined as a named vector or a data frame. Although it is not mandatory for the train_magmaclust function to run, gradients be can provided within kernel function definition. See for example se_kernel to create a custom kernel function displaying an adequate format to be used in MagmaClust.

Value

A list, containing the results of the VEM algorithm used in the training step of MagmaClust. The elements of the list are:

- hp_k: A tibble containing the trained hyper-parameters for the mean process' kernel and the mixture proportions for each cluster.
- hp_i: A tibble containing the trained hyper-parameters for the individual processes' kernels.
- hyperpost: A sub-list containing the parameters of the mean processes' hyper-posterior distribution, namely:
  - mean: A list of tibbles containing, for each cluster, the hyper-posterior mean parameters evaluated at each Input.
  - cov: A list of matrices containing, for each cluster, the hyper-posterior covariance parameter of the mean process.
  - mixture: A tibble, indicating the mixture probabilities in each cluster for each individual.
- ini_args: A list containing the initial function arguments and values for the hyper-prior means, the hyper-parameters. In particular, if those arguments were set to NULL, ini_args allows us to retrieve the (randomly chosen) initialisations used during training.
- seq_elbo: A vector, containing the sequence of ELBO values associated with each iteration.
- converged: A logical value indicated whether the algorithm converged.
- training_time: Total running time of the complete training.

Examples

```
TRUE
```

update_mixture *Update the mixture probabilities for each individual and each cluster*

Description

Update the mixture probabilities for each individual and each cluster

Usage

```
update_mixture(db, mean_k, cov_k, hp, kern, prop_mixture, pen_diag)
```

Arguments

db	A tibble or data frame. Columns required: ID, Input, Output. Additional columns for covariates can be specified.
mean_k	A list of the K hyper-posterior mean parameters.
cov_k	A list of the K hyper-posterior covariance matrices.
hp	A named vector, tibble or data frame of hyper-parameters associated with kern, the individual process' kernel. The columns/elements should be named according to the hyper-parameters that are used in kern.
kern	A kernel function, defining the covariance structure of the individual GPs.
prop_mixture	A tibble containing the hyper-parameters associated with each individual, indicating in which cluster it belongs.
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

Compute the hyper-posterior multinomial distributions by updating mixture probabilities.

Examples

```
TRUE
```

ve_step *E-Step of the VEM algorithm*

Description

Expectation step of the Variational EM algorithm used to compute the parameters of the hyper-posteriors distributions for the mean processes and mixture variables involved in MagmaClust.

Usage

```
ve_step(db, m_k, kern_k, kern_i, hp_k, hp_i, old_mixture, pen_diag)
```

Arguments

db	A tibble or data frame. Columns required: ID, Input, Output. Additional columns for covariates can be specified.
m_k	A named list of vectors, corresponding to the prior mean parameters of the K mean GPs.
kern_k	A kernel function, associated with the K mean GPs.
kern_i	A kernel function, associated with the M individual GPs.
hp_k	A named vector, tibble or data frame of hyper-parameters associated with kern_k.
hp_i	A named vector, tibble or data frame of hyper-parameters associated with kern_i.
old_mixture	A list of mixture values from the previous iteration.
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A named list, containing the elements mean, a tibble containing the Input and associated Output of the hyper-posterior mean parameters, cov, the hyper-posterior covariance matrices, and mixture, the probabilities to belong to each cluster for each individual.

Examples

```
TRUE
```

---

vm_step *V-Step of the VEM algorithm*

Description

Maximization step of the Variational EM algorithm used to compute hyper-parameters of all the kernels involved in MagmaClust.

Usage

```
vm_step(
  db,
  old_hp_k,
  old_hp_i,
  list_mu_param,
  kern_k,
  kern_i,
  m_k,
  common_hp_k,
  common_hp_i,
  pen_diag
)
```

Arguments

db	A tibble or data frame. Columns required: ID, Input, Output. Additional columns for covariates can be specified.
old_hp_k	A named vector, tibble or data frame, containing the hyper-parameters from the previous M-step (or initialisation) associated with the mean GPs.
old_hp_i	A named vector, tibble or data frame, containing the hyper-parameters from the previous M-step (or initialisation) associated with the individual GPs.
list_mu_param	List of parameters of the K mean GPs.
kern_k	A kernel used to compute the covariance matrix of the mean GP at corresponding timestamps.
kern_i	A kernel used to compute the covariance matrix of individuals GP at corresponding timestamps.
m_k	A named list of prior mean parameters for the K mean GPs. Length = 1 or nrow(unique(db$Input))
common_hp_k	A boolean indicating whether hp are common among mean GPs (for each mu_k)
common_hp_i	A boolean indicating whether hp are common among individual GPs (for each y_i)
pen_diag	A number. A jitter term, added on the diagonal to prevent numerical issues when inverting nearly singular matrices.

Value

A named list, containing the elements hp_k, a tibble containing the hyper-parameters associated with each cluster, hp_i, a tibble containing the hyper-parameters associated with the individual GPs, and prop_mixture_k, a tibble containing the hyper-parameters associated with each individual, indicating the probabilities to belong to each cluster.

Examples

```
TRUE
```

Index

chol_inv_jitter, 3

data_allocate_cluster, 4, *37*
dmnorm, 4
draw, 5

e_step, 9
elbo_clust_multi_GP, 5
elbo_clust_multi_GP_common_hp_i, 6
elbo_GP_mod_common_hp_k, 7
elbo_monitoring_VEM, 8

gr_clust_multi_GP, 9
gr_clust_multi_GP_common_hp_i, 10
gr_GP, 11
gr_GP_mod, 12
gr_GP_mod_common_hp, 12
gr_GP_mod_common_hp_k, 13
gr_sum_logL_GP_clust, 14

hp, 15
hyperposterior, 16, *56*
hyperposterior_clust, 18

ini_kmeans, 20
ini_mixture, 20, *62*

kern_to_cov, 21
kern_to_inv, 22

lin_kernel, 23
list_kern_to_cov, 24
list_kern_to_inv, 25
logL_GP, 25
logL_GP_mod, 26
logL_GP_mod_common_hp, 27
logL_monitoring, 28

m_step, 30
MagmaClustR, 29

perio_kernel, 31
plot_db, 32
plot_gif, 32, *40*
plot_gp, 34
plot_magmaclust, 36
pred_gif, *33*, 38
pred_gp, *35*, 40, *48*
pred_magma, *16*, *35*, *39*, 42, *43*, *48*
pred_magmaclust, *18*, *36, 37*, 44, *45*, *57*
proba_max_cluster, 46

rq_kernel, 47

sample_gp, 47
se_kernel, 50, *60*, *63*
select_nb_cluster, 49
simu_db, 51
simu_indiv_se, 53
sum_logL_GP_clust, 53

train_gp, *39*, *41, 42*, 54
train_gp_clust, *44, 45*, 56
train_magma, *38, 39*, *42, 43*, *56*, 58
train_magmaclust, *4*, *37*, *44, 45*, *50*, *57*, 61
transition_states, *34*

update_mixture, 64

ve_step, 64
vm_step, 65