Package: MagmaClustR (via r-universe)

August 27, 2024

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

Version 1.2.1

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 (2023) <https://jmlr.org/papers/v24/20-1321.html>. 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, plyr, purrr, Rcpp, rlang, stats, tibble, tidyr, tidyselect

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

LinkingTo Rcpp

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Depends R (>= 2.10)

Repository https://arthurleroy.r-universe.dev

RemoteUrl https://github.com/arthurleroy/magmaclustr

RemoteRef HEAD

RemoteSha 1ab2f011af0265acc05efb38ae8857f4046bf71b

Contents

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
```

expand_grid_inputs *Expand a grid of inputs*

Description

Expand a grid of inputs

Usage

```
expand_grid_inputs(Input, ...)
```

Arguments

`Input`	A vector of inputs.
`...`	As many vector of covariates as desired. We advise to give explicit names when using the function.

Value

A tibble containing all the combination of values of the parameters.

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 individuals.

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

Arguments

`trained_model`	A list, containing the information coming from a Magma model, previously trained using the `train_magma` function. If `trained_model` is not provided, the arguments `data`, `hp_0`, `hp_i`, `kern_0`, and `kern_i` are all required.
`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'. Recovered from `trained_model` if not provided.
`hp_0`	A named vector, tibble or data frame of hyper-parameters associated with `kern_0`. Recovered from `trained_model` if not provided.

`hp_i` A tibble or data frame of hyper-parameters associated with `kern_i`. Recovered from `trained_model` if not provided.

`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). Recovered from `trained_model` if not provided.

`kern_i` A kernel function, associated with the individual GPs. ("SE", "PERIO" and "RQ" are aso available here). Recovered from `trained_model` if not provided.

`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 or a data frame, 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 the user might want to use `hyperpost_clust` for a tailored control of the prediction procedure.

Usage

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

Arguments

`trained_model`	A list, containing the information coming from a Magma model, previously trained using the `train_magma` function. If `trained_model` is not provided, the arguments `data`, `mixture`, `hp_k`, `hp_i`, `kern_k`, and `kern_i` are all required.
`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`. Recovered from `trained_model` if not provided.
`mixture`	A tibble or data frame, indicating the mixture probabilities of each cluster for each individual. Required column: `ID`. Recovered from `trained_model` if not provided.
`hp_k`	A tibble or data frame of hyper-parameters associated with `kern_k`. Recovered from `trained_model` if not provided.

`hp_i` A tibble or data frame of hyper-parameters associated with `kern_i`. Recovered from `trained_model` if not provided.

`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). Recovered from `trained_model` if not provided.

`kern_i` A kernel function, associated with the individual GPs. ("SE", "LIN", PERIO" and "RQ" are also available here). Recovered from `trained_model` if not provided.

`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 or a data frame, 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
```

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. *Journal of Machine Learning Research*, 2023, https://jmlr.org/papers/v24/20-1321.html

Examples

Simulate a dataset, train and predict with Magma :

```
set.seed(4242)
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(7)

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(4242)
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))
```

Author(s)

Maintainer: Arthur Leroy <arthur.leroy.pro@gmail.com> (ORCID)

Authors:

- Pierre Latouche <pierre.latouche@gmail.com>

Other contributors:

- Pierre Pathé <pathepierre@gmail.com> [contributor]
- Alexia Grenouillat <grenouil@insa-toulouse.fr> [contributor]
- Hugo Lelievre <lelievre@insa-toulouse.fr> [contributor]

See Also

Useful links:

- https://github.com/ArthurLeroy/MagmaClustR
- https://arthurleroy.github.io/MagmaClustR/
- Report bugs at https://github.com/ArthurLeroy/MagmaClustR/issues

---

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,
  samples = FALSE,
  nb_samples = 50,
  plot_mean = TRUE,
  alpha_samples = 0.3,
  prob_CI = 0.95,
  size_data = 3,
  size_data_train = 1,
  alpha_data_train = 0.5
)

plot_magma(
  pred_gp,
  x_input = NULL,
  data = NULL,
  data_train = NULL,
  prior_mean = NULL,
  y_grid = NULL,
  heatmap = FALSE,
  samples = FALSE,
  nb_samples = 50,
  plot_mean = TRUE,
  alpha_samples = 0.3,
  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.
samples	A logical value indicating whether the GP prediction should be represented as a collection of samples drawn from the posterior. If FALSE (default), the mean curve and associated Credible Interval are displayed.
nb_samples	A number, indicating the number of samples to be drawn from the predictive posterior distribution. For two-dimensional graphs, only one sample can be displayed.
plot_mean	A logical value, indicating whether the mean prediction should be displayed on the graph when samples = TRUE.
alpha_samples	A number, controlling transparency of the sample curves.
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, as a collection of samples drawn from the posterior if `samples` = TRUE, 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,
  samples = FALSE,
  nb_samples = 50,
  plot_mean = TRUE,
  alpha_samples = 0.3,
  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 mixture 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.
samples	A logical value indicating whether the GP mixture should be represented as a collection of samples drawn from the posterior. If FALSE (default), the mean curve (and associated Credible Interval if available) are displayed.
nb_samples	A number, indicating the number of samples to be drawn from the predictive posterior distribution. For two-dimensional graphs, only one sample can be displayed.

plot_mean	A logical value, indicating whether the mean prediction should be displayed on the graph when samples = TRUE.
alpha_samples	A number, controlling transparency of the sample curves.
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
```

---

plot_samples *Display realisations from a (mixture of) GP prediction*

Description

Display samples drawn from the posterior of a GP, Magma or MagmaClust prediction. According to the dimension of the inputs, the graph may represent curves or a heatmap.

Usage

```
plot_samples(
  pred = NULL,
  samples = NULL,
  nb_samples = 50,
  x_input = NULL,
  plot_mean = TRUE,
  alpha_samples = 0.3
)
```

Arguments

pred	A list, typically coming from pred_gp, pred_magma or pred_magmaclust functions, using the argument 'get_full_cov = TRUE'. Required elements: pred, cov. This argument is needed if samples is missing.
samples	A tibble or data frame, containing the samples generated from a GP, Magma, or MagmaClust prediction. Required columns: Input, Sample, Output. This argument is needed if pred is missing.
nb_samples	A number, indicating the number of samples to be drawn from the predictive posterior distribution. For two-dimensional graphs, only one sample can be displayed.
x_input	A vector of character strings, indicating which 'column' should be displayed in the case of multidimensional inputs. 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 the y-axis.
plot_mean	A logical value, indicating whether the mean prediction should be displayed on the graph.
alpha_samples	A number, controlling transparency of the sample curves.

Value

Graph of samples drawn from a posterior distribution of a GP, Magma, or MagmaClust prediction.

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,
  grid_inputs = NULL,
  hyperpost = NULL,
  mean = NULL,
  hp = NULL,
```

```
  kern = "SE",
  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.
`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 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).

`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 standard 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 = NULL,
  grid_inputs = NULL,
  mean = NULL,
  hp = NULL,
  kern = "SE",
  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'. If NULL, the prior GP is returned.
`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`.
`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 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).
`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 = NULL,
  trained_model = NULL,
  grid_inputs = NULL,
  hp = NULL,
  kern = "SE",
  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'. If NULL, the mean process from trained_model is returned as a generic prediction.

`trained_model` A list, containing the information coming from a Magma model, previously trained using the `train_magma` function.

`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`.

`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).

`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 = NULL,
  trained_model = NULL,
  grid_inputs = NULL,
  mixture = NULL,
  hp = NULL,
  kern = "SE",
  hyperpost = NULL,
  prop_mixture = NULL,
  get_hyperpost = FALSE,
  get_full_cov = TRUE,
  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'. If NULL, the mixture of mean processes from `trained_model` is returned as a generic prediction.

`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.

`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`.

`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).

`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
```

`regularize_data` *Regularise a grid of inputs in a dataset*

Description

Modify the original grid of inputs to make it more 'regular' (in the sense that the interval between each observation is constant, or corresponds to a specific pattern defined by the user). In particular, this function can also be used to summarise several data points into one, at a specific location. In this case, the output values are averaged according to the 'summarise_fct' argument.

Usage

```
regularize_data(
  data,
  size_grid = 30,
  grid_inputs = NULL,
  summarise_fct = base::mean
)

regularise_data(
  data,
  size_grid = 30,
  grid_inputs = NULL,
  summarise_fct = base::mean
)
```

Arguments

`data`	A tibble or data frame. Required columns: `ID`, `Input` `Output`. The `ID` column contains the unique names/codes used to identify each individual/task (or batch of data). The `Input` column corresponds to observed locations (an explanatory variable). The `Output` column specifies the associated observed values (the response variable). The data frame can also provide as many additional inputs as desired, with no constraints on the column names.
`size_grid`	An integer, which indicates the number of equispaced points each column must contain. Each original input value will be collapsed to the closest point of the new regular grid, and the associated outputs are averaged using the 'summarise_fct' function. This argument is used when 'grid_inputs' is left to 'NULL'. Default value is 30.

grid_inputs	A data frame, corresponding to a pre-defined grid of inputs according to which we want to regularise a dataset. Column names must be similar to those appearing in data. If NULL (default), a default grid of inputs is defined: for each input column in data, a regular sequence is created from the min to the max values, with a number of equispaced points being equal to the 'size_grid' argument.
summarise_fct	A character string or a function. If several similar inputs are associated with different outputs, the user can choose the summarising function for the output among the following: min, max, mean, median. A custom function can be defined if necessary. Default is "mean".

Value

A data frame, where input columns have been regularised as desired.

Examples

```
data = tibble::tibble(ID = 1, Input = 0:100, Output = -50:50)

## Define a 1D input grid of 10 points
regularize_data(data, size_grid = 10)

## Define a 1D custom grid
my_grid = tibble::tibble(Input = c(5, 10, 25, 50, 100))
regularize_data(data, grid_inputs = my_grid)

## Define a 2D input grid of 5x5 points
data_2D = cbind(ID = 1, expand.grid(Input=1:10, Input2=1:10), Output = 1:100)
regularize_data(data_2D, size_grid = 5)

## Define a 2D custom input grid
my_grid_2D = MagmaClustR::expand_grid_inputs(c(2, 4, 8), 'Input2' = c(3, 5))
regularize_data(data_2D, grid_inputs = my_grid_2D)
```

---

sample_gp *Draw samples from a posterior GP/Magma distribution*

Description

Draw samples from a posterior GP/Magma distribution

Usage

```
sample_gp(pred_gp, nb_samples = 50)

sample_magma(pred_gp, nb_samples = 50)
```

Arguments

pred_gp	A list, typically coming from pred_magma or pred_gp functions, with argument 'get_full_cov = TRUE'. Required elements: pred, cov.
nb_samples	A number, indicating the number of samples to be drawn from the predictive posterior distribution. For two-dimensional graphs, only one sample can be displayed.

Value

A tibble or data frame, containing the samples generated from a GP prediction. Format: Input, Sample, Output.

Examples

```
TRUE
```

---

sample_magmaclust *Draw samples from a MagmaClust posterior distribution*

Description

Draw samples from a MagmaClust posterior distribution

Usage

```
sample_magmaclust(pred_clust, nb_samples = 50)
```

Arguments

pred_clust	A list, typically coming from pred_magmaclust, with argument get_full_cov = TRUE'. Required elements: pred, cov, mixture.
nb_samples	A number, indicating the number of samples to be drawn from the predictive posterior distribution. For two-dimensional graphs, only one sample can be displayed.

Value

A tibble or data frame, containing the samples generated from a GP prediction. Format: Cluster, Proba, Input, Sample, Output.

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. The hp function can be used to draw custom hyper-parameters with the correct format.
ini_hp_i	A tibble or data frame of hyper-parameters associated with kern_i. The hp function can be used to draw custom hyper-parameters with the correct format.db
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
```

---

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),
  grid_cov = seq(0, 10, 0.5),
  common_input = TRUE,
```

```
  common_hp = TRUE,
  add_hp = FALSE,
  add_clust = FALSE,
  int_mu_v = c(4, 5),
  int_mu_l = c(0, 1),
  int_i_v = c(1, 2),
  int_i_l = c(0, 1),
  int_i_sigma = c(0, 0.2),
  lambda_int = c(30, 40),
  m_int = c(0, 10),
  lengthscale_int = c(30, 40),
  m0_slope = c(-5, 5),
  m0_intercept = c(-50, 50)
)
```

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).
grid_cov	A vector of numbers defining a grid of observations (i.e. the covariate 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.

lambda_int	A vector of 2 numbers, defining an interval of admissible values for the lambda parameter of the 2D exponential.
m_int	A vector of 2 numbers, defining an interval of admissible values for the mean of the 2D exponential.
lengthscale_int	A vector of 2 numbers, defining an interval of admissible values for the length-scale parameter of the 2D exponential.
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.

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, add_clust = TRUE)
```

---

swimmers *French swimmers performances data on 100m freestyle events*

Description

A subset of data from reported performances of French swimmers during 100m freestyle competitions between 2002 and 2016. See https://link.springer.com/article/10.1007/s10994-022-06172-1 and https://www.mdpi.com/2076-3417/8/10/1766 for dedicated description and analysis.

Usage

```
swimmers
```

Format

`swimmers`:
A data frame with 76,832 rows and 4 columns:

ID Indentifying number associated to each swimmer

Input Age in years

Output Performance in seconds on a 100m freestyle event

Gender Competition gender

Source

https://ffn.extranat.fr/webffn/competitions.php?idact=nat

`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. The `hp` function can be used to draw custom hyper-parameters with the correct format.

`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. The `hp` function can be used to draw custom hyper-parameters with the correct format.
`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).

`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. The hp function can be used to draw custom hyper-parameters with the correct format.
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. The hp function can be used to draw custom hyper-parameters with the correct format.

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`. The `hp` function can be used to draw custom hyper-parameters with the correct format.

`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`. The `hp` function can be used to draw custom hyper-parameters with the correct format.

`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
```

---

weight *Weight follow-up data of children in Singapore*

Description

A subset of data from the GUSTO project (https://www.gusto.sg/) collecting the weight over time of several children in Singapore. See https://arxiv.org/abs/2011.07866 for dedicated description and analysis.

Usage

```
weight
```

Format

`weight`**:**
A data frame with 3,629 rows and 4 columns:

ID Indentifying number associated to each child

sex Biological gender

Input Age in months

Output Weight in kilograms

Source

https://www.gusto.sg/

Index