Package 'dgpsi'

October 13, 2022

Type Package

Title Interface to 'dgpsi' for Deep and Linked Gaussian Process Emulations

Version 2.1.5

Maintainer Deyu Ming <deyu.ming.16@ucl.ac.uk>

Description Interface to the 'python' package 'dgpsi' for Gaussian process, deep Gaussian process, and linked Gaussian process emulations of computer models and systems of computer models. The implementations follow Ming & Guillas (2021) <doi:10.1137/20M1323771> and Ming, Williamson, & Guillas (2022) <arXiv:2107.01590>. To get started with the package, see <https://mingdeyu.github.io/dgpsi-R/>.

License MIT + file LICENSE

URL https://github.com/mingdeyu/dgpsi-R,
https://mingdeyu.github.io/dgpsi-R/

BugReports https://github.com/mingdeyu/dgpsi-R/issues

Encoding UTF-8

Depends R (>= 4.0)

Imports reticulate (>= 1.26), benchmarkme (>= 1.0.8), utils, ggplot2, reshape2, patchwork

Suggests knitr, rmarkdown, MASS, R.utils, spelling

VignetteBuilder knitr

RoxygenNote 7.2.1

Language en-US

NeedsCompilation no

Author Deyu Ming [aut, cre, cph],
Daniel Williamson [aut]

Repository CRAN

Date/Publication 2022-09-29 12:50:02 UTC

R topics documented:

`combine` *Combine layers*

Description

This function combines customized layers into a DGP or linked (D)GP structure.

Usage

```
combine(...)
```

Arguments

`...`	a sequence of lists: 1. For DGP emulations, each list represents a DGP layer and contains GP nodes (produced by `kernel()`), or likelihood nodes (produced by `Poisson()`, `Hetero()`, or `NegBin()`). 2. For linked (D)GP emulations, each list represents a system layer and contains emulators (produced by `gp()` or `dgp()`) in that layer.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

A list defining a DGP structure (for `struc` of `dgp()`) or a linked (D)GP structure (for `struc` for `lgp()`).

Examples

```
## Not run:

# See lgp() for an example.

## End(Not run)
```

`continue` *Continue the training of a DGP emulator*

Description

This function implements additional training iterations for a DGP emulator.

Usage

```
continue(object, N = 500, ess_burn = 10, verb = TRUE, burnin = NULL, B = 50)
```

Arguments

`object`	an instance of the dgp class.
`N`	additional number of iterations for the DGP emulator training. Defaults to `500`.
`ess_burn`	number of burnin steps for the ESS-within-Gibbs at each I-step of the training. Defaults to `10`.
`verb`	a bool indicating if the progress bar will be printed during the training: 1. `FALSE`: the training progress bar will not be displayed. 2. `TRUE`: the training progress bar will be displayed. Defaults to `TRUE`.
`burnin`	the number of training iterations to be discarded for point estimates calculation. Must be smaller than the overall training iterations so-far implemented. If this is not specified, only the last 25% of iterations are used. This overrides the value of `burnin` set in `dgp()`. Defaults to `NULL`.
`B`	the number of imputations to produce the predictions. Increase the value to account for more imputation uncertainties. This overrides the value of `B` set in `dgp()`. Defaults to `50`.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

An updated `object`.

Note

One can also use this function to fit an untrained DGP emulator constructed by `dgp()` with `training = FALSE`.

Examples

```
## Not run:

# See dgp() for an example.

## End(Not run)
```

dgp — *Deep Gaussian process emulator construction*

Description

This function builds and trains a DGP emulator.

Usage

```
dgp(
  X,
  Y,
  struc = NULL,
  depth = 2,
  name = "sexp",
  lengthscale = 1,
  share = TRUE,
  nugget_est = FALSE,
  nugget = 1e-06,
  connect = TRUE,
  likelihood = NULL,
  training = TRUE,
  verb = TRUE,
  check_rep = TRUE,
  rff = FALSE,
  M = NULL,
  N = 500,
  ess_burn = 10,
  burnin = NULL,
  B = 50,
  internal_input_idx = NULL,
  linked_idx = NULL
)
```

Arguments

`X`	a matrix where each row is an input training data point and each column is an input dimension.
`Y`	a matrix containing observed training output data. The matrix has its rows being output data points and columns being output dimensions. When `likelihood` (see below) is not `NULL`, `Y` must be a matrix with only one column.
`struc`	a list that specifies a user-defined DGP structure. It should contain L (the number of DGP layers) sub-lists, each of which represents a layer and contains a number of GP nodes (defined by `kernel()`) in the corresponding layer. The final layer of the DGP structure (i.e., the final sub-list in `struc`) can be a likelihood layer that contains a likelihood function (e.g., `Poisson()`). When `struc = NULL`, the DGP structure is automatically generated and can be checked by applying `summary()` to the output from `dgp()` with `training = FALSE`. If this argument is used (i.e., user provides a customized DGP structure), arguments `depth`, `name`, `lengthscale`, `nugget_est`, `nugget`, `connect`, `likelihood`, and `internal_input_idx` will NOT be used. Defaults to `NULL`.
`depth`	number of layers (including the likelihood layer) for a DGP structure. `depth` must be at least `2`. Defaults to `2`. This argument is only used when `struc = NULL`.
`name`	kernel function to be used. Either `"sexp"` for squared exponential kernel or `"matern2.5"` for Matérn-2.5 kernel. Defaults to `"sexp"`. This argument is only used when `struc = NULL`.
`lengthscale`	initial lengthscales for GP nodes in the DGP emulator. It can be a single numeric value or a vector: 1. if it is a single numeric value, the value will be applied as the initial lengthscales for all GP nodes in the DGP hierarchy. 2. if it is a vector, each element of the vector specifies the initial lengthscales that will be applied to all GP nodes in the corresponding layer. The vector should have a length of `depth` if `likelihood = NULL` or a length of `depth - 1` if `likelihood` is not `NULL`. Defaults to a numeric value of `1.0`. This argument is only used when `struc = NULL`.
`share`	a bool indicating if all input dimensions of a GP node share a common lengthscale. Defaults to `TRUE`. This argument is only used when `struc = NULL`.
`nugget_est`	a bool or a bool vector that indicates if the nuggets of GP nodes (if any) in the final layer are to be estimated. If a single bool is provided, it will be applied to all GP nodes (if any) in the final layer. If a bool vector (which must have a length of `ncol(Y)`) is provided, each bool element in the vector will be applied to the corresponding GP node (if any) in the final layer. The value of a bool has following effects: • `FALSE`: the nugget of the corresponding GP in the final layer is fixed to the corresponding value defined in `nugget` (see below). • `TRUE`: the nugget of the corresponding GP in the final layer will be estimated with the initial value given by the correspondence in `nugget` (see below).

	Defaults to `FALSE`. This argument is only used when `struc = NULL`.
`nugget`	the initial nugget value(s) of GP nodes (if any) in the final layer. If it is a single numeric value, it will be applied to all GP nodes (if any) in the final layer. If it is a vector (which must have a length of `ncol(Y)`), each numeric in the vector will be applied to the corresponding GP node (if any) in the final layer. Set `nugget` to a small value and the corresponding bool in `nugget_est` to `FASLE` for deterministic emulations where the emulator interpolates the training data points. Set `nugget` to a reasonable larger value and the corresponding bool in `nugget_est` to `TRUE` for stochastic emulations where the computer model outputs are assumed to follow a homogeneous Gaussian distribution. Defaults to `1e-6`. This argument is only used when `struc = NULL`.
`connect`	a bool indicating whether to implement global input connection to the DGP structure. Defaults to `TRUE`. This argument is only used when `struc = NULL`.
`likelihood`	the likelihood type of a DGP emulator: 1. `NULL`: no likelihood layer is included in the emulator. 2. `"Hetero"`: a heteroskedastic Gaussian likelihood layer is added for stochastic emulation where the computer model outputs are assumed to follow a heteroskedastic Gaussian distribution (i.e., the computer model outputs have varying noises). 3. `"Poisson"`: a Poisson likelihood layer is added for stochastic emulation where the computer model outputs are assumed to a Poisson distribution. 4. `"NegBin"`: a negative Binomial likelihood layer is added for stochastic emulation where the computer model outputs are assumed to follow a negative Binomial distribution. When `likelihood` is not `NULL`, the values of `nugget_est` and `nugget` are overridden by `FALSE` and `1e-6` respectively. Defaults to `NULL`. This argument is only used when `struc = NULL`.
`training`	a bool indicating if the initialized DGP emulator will be trained. When set to `FALSE`, `dgp()` returns an untrained DGP emulator, to which one can apply `summary()` to inspect its specifications (especially when a customized `struc` is provided) or apply `predict()` to check its emulation performance before the training. Defaults to `TRUE`.
`verb`	a bool indicating if the trace information on DGP emulator construction and training will be printed during the function execution. Defaults to `TRUE`.
`check_rep`	a bool indicating whether to check the repetitions in the dataset, i.e., if one input position has multiple outputs. Defaults to `TRUE`.
`rff`	a bool indicating whether to use random Fourier features to approximate the correlation matrices in training. Turning on this option could help accelerate the training when the training data is relatively large but may reduce the quality of the resulting emulator. Defaults to `FALSE`.
`M`	the number of features to be used by random Fourier approximation. It is only used when `rff` is set to `TRUE`. Defaults to `NULL`. If it is `NULL`, `M` is automatically set to `max(100, ceiling(sqrt(nrow(X))*log(nrow(X))))`.
`N`	number of iterations for the training. Defaults to `500`. This argument is only used when `training = TRUE`.

`ess_burn` number of burnin steps for the ESS-within-Gibbs at each I-step of the training. Defaults to `10`. This argument is only used when `training = TRUE`.

`burnin` the number of training iterations to be discarded for point estimates of model parameters. Must be smaller than the training iterations `N`. If this is not specified, only the last 25% of iterations are used. Defaults to `NULL`. This argument is only used when `training = TRUE`.

`B` the number of imputations to produce the later predictions. Increase the value to account for more imputation uncertainties. Decrease the value for lower imputation uncertainties but faster predictions. Defaults to `50`.

`internal_input_idx`
column indices of `X` that are generated by the linked emulators in the preceding layers. Set `internal_input_idx = NULL` if the DGP emulator is in the first layer of a system or all columns in `X` are generated by the linked emulators in the preceding layers. Defaults to `NULL`. This argument is only used when `struc = NULL`.

`linked_idx` either a vector or a list of vectors:

- If `linked_idx` is a vector, it gives indices of columns in the pooled output matrix (formed by column-combined outputs of all emulators in the feeding layer) that feed into the DGP emulator. If the DGP emulator is in the first layer of a linked emulator system, the vector gives the column indices of the global input (formed by column-combining all input matrices of emulators in the first layer) that the DGP emulator will use. The length of the vector shall equal to the length of `internal_input_idx` when `internal_input_idx` is not `NULL`.
- When the DGP emulator is not in the first layer of a linked emulator system, `linked_idx` can be a list that gives the information on connections between the DGP emulator and emulators in all preceding layers. The length of the list should equal to the number of layers before the DGP emulator. Each element of the list is a vector that gives indices of columns in the pooled output matrix (formed by column-combined outputs of all emulators) in the corresponding layer that feed into the DGP emulator. If the DGP emulator has no connections to any emulator in a certain layer, set `NULL` in the corresponding position of the list. The order of input dimensions in `X[,internal_input_idx]` should be consistent with `linked_idx`. For example, a DGP emulator in the 4th-layer that is fed by the output dimension 2 and 4 of emulators in layer 2 and all output dimension 1 to 3 of emulators in layer 3 should have `linked_idx = list( NULL, c(2,4), c(1,2,3) )`. In addition, the first and second columns of `X[,internal_input_idx]` should correspond to the output dimensions 2 and 4 from layer 2, and the third to fifth columns of `X[,internal_input_idx]` should correspond to the output dimensions 1 to 3 from layer 3.

Set `linked_idx = NULL` if the DGP emulator will not be used for linked emulations. However, if this is no longer the case, one can use `set_linked_idx()` to add linking information to the DGP emulator. Defaults to `NULL`.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/` and learn how to customize a DGP structure.

Value

An S3 class named dgp that contains three slots:

- `constructor_obj`: a 'python' object that stores the information of the constructed DGP emulator.
- `container_obj`: a 'python' object that stores the information for the linked emulation.
- `emulator_obj`: a 'python' object that stores the information for the predictions from the DGP emulator.

The returned dgp object can be used by

- `predict()` for DGP predictions.
- `continue()` for additional DGP training iterations.
- `validate()` for LOO and OOS validations.
- `plot()` for validation plots.
- `lgp()` for linked (D)GP emulator constructions.

Note

Any R vector detected in `X` and `Y` will be treated as a column vector and automatically converted into a single-column R matrix.

Examples

```
## Not run:

# load the package and the Python env
library(dgpsi)
init_py()

# construct a step function
f <- function(x) {
  if (x < 0.5) return(-1)
  if (x >= 0.5) return(1)
  }

# generate training data
X <- seq(0, 1, length = 10)
Y <- sapply(X, f)

# training a 3-layered DGP emulator
m <- dgp(X, Y, depth = 3)

# continue for further training iterations
```

```
m <- continue(m)

# summarizing
summary(m)

# trace plot
trace_plot(m)

# LOO cross validation
m <- validate(m)
plot(m)

# prediction
test_x <- seq(0, 1, length = 200)
m <- predict(m, x = test_x)

# OOS validation
validate_x <- sample(test_x, 10)
validate_y <- sapply(validate_x, f)
plot(m, validate_x, validate_y)

# write and read the constructed emulator
write(m, 'step_dgp')
m <- read('step_dgp')

## End(Not run)
```

---

gp *Gaussian process emulator construction*

Description

This function builds and trains a GP emulator.

Usage

```
gp(
  X,
  Y,
  struc = NULL,
  name = "sexp",
  lengthscale = rep(0.2, ncol(X)),
  nugget_est = FALSE,
  nugget = 1e-06,
  training = TRUE,
  verb = TRUE,
  internal_input_idx = NULL,
  linked_idx = NULL
)
```

Arguments

`X`	a matrix where each row is an input data point and each column is an input dimension.
`Y`	a matrix with only one column and each row being an output data point.
`struc`	an object produced by `kernel()` that gives a user-defined GP specifications. When `struc = NULL`, the GP specifications are automatically generated using information provided in `name`, `lengthscale`, `nugget_est`, `nugget`, and `internal_input_idx`. Defaults to `NULL`.
`name`	kernel function to be used. Either `"sexp"` for squared exponential kernel or `"matern2.5"` for Matérn-2.5 kernel. Defaults to `"sexp"`. This argument is only used when `struc = NULL`.
`lengthscale`	initial values of lengthscales in the kernel function. It can be a single numeric value or a vector: • if it is a single numeric value, it is assumed that kernel functions across input dimensions share the same lengthscale; • if it is a vector (which must have a length of `ncol(X)`), it is assumed that kernel functions across input dimensions have different lengthscales. Defaults to a vector of 0.2. This argument is only used when `struc = NULL`.
`nugget_est`	a bool indicating if the nugget term is to be estimated: 1. `FALSE`: the nugget term is fixed to `nugget`. 2. `TRUE`: the nugget term will be estimated. Defaults to `FALSE`. This argument is only used when `struc = NULL`.
`nugget`	the initial nugget value. If `nugget_est = FALSE`, the assigned value is fixed during the training. Set `nugget` to a small value (e.g., `1e-6`) and the corresponding bool in `nugget_est` to `FASLE` for deterministic emulations where the emulator interpolates the training data points. Set `nugget` to a reasonable larger value and the corresponding bool in `nugget_est` to `TRUE` for stochastic emulations where the computer model outputs are assumed to follow a homogeneous Gaussian distribution. Defaults to `1e-6`. This argument is only used when `struc = NULL`.
`training`	a bool indicating if the initialized GP emulator will be trained. When set to `FALSE`, `gp()` returns an untrained GP emulator, to which one can apply `summary()` to inspect its specifications (especially when a customized `struc` is provided) or apply `predict()` to check its emulation performance before the training. Defaults to `TRUE`.
`verb`	a bool indicating if the trace information on GP emulator construction and training will be printed during the function execution. Defaults to `TRUE`.
`internal_input_idx`	the column indices of `X` that are generated by the linked emulators in the preceding layers. Set `internal_input_idx = NULL` if the GP emulator is in the first layer of a system or all columns in `X` are generated by the linked emulators in the preceding layers. Defaults to `NULL`. This argument is only used when `struc = NULL`.
`linked_idx`	either a vector or a list of vectors:

- If `linked_idx` is a vector, it gives indices of columns in the pooled output matrix (formed by column-combined outputs of all emulators in the feeding layer) that feed into the GP emulator. If the GP emulator is in the first layer of a linked emulator system, the vector gives the column indices of the global input (formed by column-combining all input matrices of emulators in the first layer) that the GP emulator will use. The length of the vector shall equal to the length of `internal_input_idx` when `internal_input_idx` is not `NULL`.
- When the GP emulator is not in the first layer of a linked emulator system, `linked_idx` can be a list that gives the information on connections between the GP emulator and emulators in all preceding layers. The length of the list should equal to the number of layers before the GP emulator. Each element of the list is a vector that gives indices of columns in the pooled output matrix (formed by column-combined outputs of all emulators) in the corresponding layer that feed into the GP emulator. If the GP emulator has no connections to any emulator in a certain layer, set `NULL` in the corresponding position of the list. The order of input dimensions in `X[,internal_input_idx]` should be consistent with `linked_idx`. For example, a GP emulator in the second layer that is fed by the output dimension 1 and 3 of emulators in layer 1 should have `linked_idx = list( c(1,3) )`. In addition, the first and second columns of `X[,internal_input_idx]` should correspond to the output dimensions 1 and 3 from layer 1.

Set `linked_idx = NULL` if the GP emulator will not be used for linked emulations. However, if this is no longer the case, one can use `set_linked_idx()` to add linking information to the GP emulator. Defaults to `NULL`.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

An S3 class named gp that contains three slots:

- `constructor_obj`: a 'python' object that stores the information of the constructed GP emulator.
- `container_obj`: a 'python' object that stores the information for the linked emulation.
- `emulator_obj`: a 'python' object that stores the information for the predictions from the GP emulator.

The returned gp object can be used by

- `predict()` for GP predictions.
- `validate()` for LOO and OOS validations.
- `plot()` for validation plots.
- `lgp()` for linked (D)GP emulator constructions.

Note

Any R vector detected in X and Y will be treated as a column vector and automatically converted into a single-column R matrix.

Examples

```
## Not run:
# load the package and the Python env
library(dgpsi)
init_py()

# construct a step function
f <- function(x) {
   if (x < 0.5) return(-1)
   if (x >= 0.5) return(1)
  }

# generate training data
X <- seq(0, 1, length = 10)
Y <- sapply(X, f)

# training
m <- gp(X, Y)

# summarizing
summary(m)

# LOO cross validation
m <- validate(m)
plot(m)

# prediction
test_x <- seq(0, 1, length = 200)
m <- predict(m, x = test_x)

# OOS validation
validate_x <- sample(test_x, 10)
validate_y <- sapply(validate_x, f)
plot(m, validate_x, validate_y)

# write and read the constructed emulator
write(m, 'step_gp')
m <- read('step_gp')

## End(Not run)
```

Hetero *Initialize a heteroskedastic Gaussian likelihood node*

Description

This function constructs a likelihood object to represent a heteroskedastic Gaussian likelihood node.

Usage

```
Hetero(input_dim = NULL)
```

Arguments

`input_dim`	a vector of length two that contains the indices of two GP nodes in the feeding layer whose outputs feed into this likelihood node. When set to `NULL`, all outputs from GP nodes in the feeding layer feed into this likelihood node, and in such a case one needs to ensure that only two GP nodes are specified in the feeding layer. Defaults to `NULL`.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

A 'python' object to represent a heteroskedastic Gaussian likelihood node.

Note

The heteroskedastic Gaussian likelihood node can only be linked to two feeding GP nodes.

Examples

```
## Not run:

# Check https://mingdeyu.github.io/dgpsi-R/ for examples
# on how to customize DGP structures using Hetero().

## End(Not run)
```

`init_py` *'python' environment initialization*

Description

This function initializes the 'python' environment for the package.

Usage

```
init_py(py_ver = NULL, dgpsi_ver = NULL)
```

Arguments

py_ver	a string that gives the 'python' version to be installed. If py_ver = NULL, the default 'python' version '3.9.13' will be installed.
dgpsi_ver	a string that gives the 'python' version of 'dgpsi' to be used. If dgpsi_ver = NULL, the latest 'python' version of 'dgpsi' will be used.

Details

See further examples and tutorials at https://mingdeyu.github.io/dgpsi-R/.

Value

No return value, called to install required 'python' environment.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

---

kernel *Initialize a Gaussian process node*

Description

This function constructs a kernel object to represent properties of a Gaussian process node.

Usage

```
kernel(
  length,
  scale = 1,
  nugget = 1e-06,
  name = "sexp",
  prior_name = "ga",
  prior_coef = c(1.6, 0.3),
  nugget_est = FALSE,
  scale_est = FALSE,
  input_dim = NULL,
  connect = NULL
)
```

Arguments

`length`	a vector of lengthscales. The length of the vector equals to: 1. either one if the lengthscales in the kernel function are assumed same across input dimensions; or 2. the total number of input dimensions, which is the sum of the number of feeding GP nodes in the last layer (defined by the argument `input_dim`) and the number of connected global input dimensions (defined by the argument `connect`), if the lengthscales in the kernel function are assumed different across input dimensions.
`scale`	the variance of a GP node. Defaults to `1`.
`nugget`	the nugget term of a GP node. Defaults to `1e-6`.
`name`	kernel function to be used. Either `"sexp"` for squared exponential kernel or `"matern2.5"` for Matérn-2.5 kernel. Defaults to `"sexp"`.
`prior_name`	prior options for the lengthscales and nugget term. Either gamma (`"ga"`) or inverse gamma (`"inv_ga"`) distribution for the lengthscales and nugget term. Set `NULL` to disable the prior. Defaults to `"ga"`.
`prior_coef`	a vector that contains two values specifying the shape and rate parameters of the gamma prior, or shape and scale parameters of the inverse gamma prior. Defaults to `c(1.6,0.3)`.
`nugget_est`	set to `TRUE` to estimate the nugget term or to `FALSE` to fix the nugget term as specified by the argument `nugget`. If set to `TRUE`, the value set to the argument `nugget` is used as the initial value. Defaults to `FALSE`.
`scale_est`	set to `TRUE` to estimate the variance (i.e., scale) or to `FALSE` to fix the variance (i.e., scale) as specified by the argument `scale`. Defaults to `FALSE`.
`input_dim`	a vector that contains either 1. the indices of GP nodes in the feeding layer whose outputs feed into this GP node; or 2. the indices of global input dimensions that are linked to the outputs of some feeding emulators, if this GP node is in the first layer of a GP or DGP, which will be used for the linked emulation. When set to `NULL`, 1. all outputs from the GP nodes in the feeding layer feed into this GP node; or 2. all global input dimensions feed into this GP node. Defaults to `NULL`.
`connect`	a vector that contains the indices of dimensions in the global input connecting to this GP node as additional input dimensions. When set to `NULL`, no global input connection is implemented. Defaults to `NULL`. When this GP node is in the first layer of a GP or DGP emulator, which will consequently be used for linked emulation, `connect` gives the indices of global input dimensions that are not connected to some feeding emulators. In such a case, set `input_dim` to a vector of indices of the remaining input dimensions that are connected to the feeding emulators.

Details

See further examples and tutorials at https://mingdeyu.github.io/dgpsi-R/.

Value

A 'python' object to represent a GP node.

Examples

```
## Not run:

# Check https://mingdeyu.github.io/dgpsi-R/ for examples
# on how to customize DGP structures using kernel().

## End(Not run)
```

---

lgp — *Linked (D)GP emulator construction*

Description

This function constructs a linked (D)GP emulator.

Usage

```
lgp(struc, B = 50)
```

Arguments

`struc`	a list contains L (the number of layers in a systems of computer models) sub-lists, each of which represents a layer and contains (D)GP emulators (represented by instances of S3 class gp or dgp) of computer models. The sub-lists are placed in the list in the same order of the specified computer model system's hierarchy.
`B`	the number of imputations to produce the predictions. Increase the value to account for more imputation uncertainties. Decrease the value for lower imputation uncertainties but faster predictions. If the system consists only GP emulators, `B` is set to `1` automatically. Defaults to `50`.

Details

See further examples and tutorials at https://mingdeyu.github.io/dgpsi-R/.

Value

An S3 class named `lgp` that contains a slot called `emulator_obj`, which is a 'python' object that stores the information for predictions from the linked emulator. The returned `lgp` object can be used by

- `predict()` for linked (D)GP predictions.
- `validate()` for the OOS validation.
- `plot()` for the validation plots.

Examples

```
## Not run:

# load the package and the Python env
library(dgpsi)
init_py()

# model 1
f1 <- function(x) {
  (sin(7.5*x)+1)/2
}
# model 2
f2 <- function(x) {
  2/3*sin(2*(2*x - 1))+4/3*exp(-30*(2*(2*x-1))^2)-1/3
}
# linked model
f12 <- function(x) {
  f2(f1(x))
}

# training data for Model 1
X1 <- seq(0, 1, length = 9)
Y1 <- sapply(X1, f1)
# training data for Model 2
X2 <- seq(0, 1, length = 13)
Y2 <- sapply(X2, f2)

# emulation of model 1
m1 <- gp(X1, Y1, name = "matern2.5", linked_idx = c(1))
# emulation of model 2
m2 <- dgp(X2, Y2, depth = 2, name = "matern2.5")
# assign linking information after the emulation construction
m2 <- set_linked_idx(m2, c(1))

# emulation of the linked model
struc <- combine(list(m1), list(m2))
m_link <- lgp(struc)

# summarizing
summary(m_link)
```

```
# prediction
test_x <- seq(0, 1, length = 300)
m_link <- predict(m_link, x = test_x)

# OOS validation
validate_x <- sample(test_x, 20)
validate_y <- sapply(validate_x, f12)
plot(m_link, validate_x, validate_y, style = 2)

# write and read the constructed linked emulator
write(m_link, 'linked_emulator')
m_link <- read('linked_emulator')

## End(Not run)
```

NegBin *Initialize a negative Binomial likelihood node*

Description

This function constructs a likelihood object to represent a negative Binomial likelihood node.

Usage

```
NegBin(input_dim = NULL)
```

Arguments

`input_dim`	a vector of length two that contains the indices of two GP nodes in the feeding layer whose outputs feed into this likelihood node. When set to `NULL`, all outputs from GP nodes in the feeding layer feed into this likelihood node, and in such a case one needs to ensure that only two GP nodes are specified in the feeding layer. Defaults to `NULL`.

Details

See further examples and tutorials at https://mingdeyu.github.io/dgpsi-R/.

Value

A 'python' object to represent a negative Binomial likelihood node.

Note

The negative Binomial likelihood node can only be linked to two feeding GP nodes.

Examples

```
## Not run:

# Check https://mingdeyu.github.io/dgpsi-R/ for examples
# on how to customize DGP structures using NegBin().

## End(Not run)
```

`nllik` *Calculate negative predicted log-likelihood*

Description

This function computes the negative predicted log-likelihood from a DGP emulator with a likelihood layer.

Usage

```
nllik(object, x, y)
```

Arguments

`object`	an instance of the dgp class and it should be produced by `dgp()` with one of the following two settings: 1. if `struc` = `NULL`, `likelihood` is not `NULL`; 2. if a customized structure is provided to `struc`, the final layer must be likelihood layer containing only one likelihood node produced by `Poisson()`, `Hetero()`, or `NegBin()`.
`x`	a matrix where each row is an input testing data point and each column is an input dimension.
`y`	a matrix with only one column where each row is a scalar-valued testing output data point.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

An updated `object` is returned with an additional slot named `NLL` that contains two elements. The first one, named `meanNLL`, is a scalar that gives the average negative predicted log-likelihood across all testing data points. The second one, named `allNLL`, is a vector that gives the negative predicted log-likelihood for each testing data point.

Note

Any R vector detected in `x` and `y` will be treated as a column vector and automatically converted into a single-column R matrix.

Examples

```
## Not run:

# Check https://mingdeyu.github.io/dgpsi-R/ for examples
# on how to compute the negative predicted log-likelihood
# using nllik().

## End(Not run)
```

---

plot *Validation plots of a constructed GP, DGP, or linked (D)GP emulator*

Description

This function draws validation plots of a GP, DGP, or linked (D)GP emulator.

Usage

```
## S3 method for class 'dgp'
plot(
  x,
  x_test = NULL,
  y_test = NULL,
  dim = NULL,
  method = "mean_var",
  style = 1,
  verb = TRUE,
  force = FALSE,
  cores = 1,
  threading = FALSE,
  ...
)

## S3 method for class 'lgp'
plot(
  x,
  x_test = NULL,
  y_test = NULL,
  dim = NULL,
  method = "mean_var",
  style = 1,
  verb = TRUE,
  force = FALSE,
  cores = 1,
  threading = FALSE,
  ...
)
```

```
## S3 method for class 'gp'
plot(
  x,
  x_test = NULL,
  y_test = NULL,
  dim = NULL,
  method = "mean_var",
  style = 1,
  verb = TRUE,
  force = FALSE,
  cores = 1,
  ...
)
```

Arguments

`x`	can be one of the following emulator classes: • the S3 class `gp`. • the S3 class `dgp`. • the S3 class `lgp`.
`x_test`	same as that of `validate()`.
`y_test`	same as that of `validate()`.
`dim`	if `dim = NULL`, the index of an emulator's input will be shown on the x-axis in validation plots. Otherwise, `dim` indicates which dimension of an emulator's input will be shown on the x-axis in validation plots: • If `x` is an instance of the `gp` of `dgp` class, `dim` is an integer. • If `x` is an instance of the `lgp` class, `dim` can be 1. an integer referring to the dimension of the global input to emulators in the first layer of a linked emulator system; or 2. a vector of three integers referring to the dimension (specified by the third integer) of the global input to an emulator (specified by the second integer) in a layer (specified by the first integer) that is not the first layer of a linked emulator system. This argument is only used when `style = 1` and the emulator input is at least two-dimensional. Defaults to `NULL`.
`method`	same as that of `validate()`.
`style`	either `1` or `2`, indicating two different types of validation plots.
`verb`	a bool indicating if the trace information on plotting will be printed during the function execution. Defaults to `TRUE`.
`force`	same as that of `validate()`.
`cores`	same as that of `validate()`.
`threading`	same as that of `validate()`.
`...`	N/A.

Details

See further examples and tutorials at https://mingdeyu.github.io/dgpsi-R/.

Value

A `patchwork` object.

Note

- `plot()` calls `validate()` internally to obtain validation results for plotting. However, `plot()` will not export the emulator object with validation results. Instead, it only returns the plotting object. For small-scale validations (i.e., small training or testing data points), direct execution of `plot()` is fine. However, for moderate- to large-scale validations, it is recommended to first run `validate()` to obtain and store validation results in the emulator object, and then supply the object to `plot()`. This is because if an emulator object has the validation results stored, each time when `plot()` is invoked, unnecessary evaluations of repetitive LOO or OOS validation will not be implemented.
- `plot()` uses information provided in `x_test` and `y_test` to produce the OOS validation plots. Therefore, if validation results are already stored in `x`, unless `x_test` and `y_test` are identical to those used by `validate()`, `plot()` will re-evaluate OOS validations before plotting.
- Any R vector detected in `x_test` and `y_test` will be treated as a column vector and automatically converted into a single-column R matrix.
- The returned `patchwork` object contains the `ggplot2` objects. One can modify the included individual ggplots by accessing them with double-bracket indexing. See https://patchwork.data-imaginist.com/ for further information.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

`Poisson` *Initialize a Poisson likelihood node*

Description

This function constructs a likelihood object to represent a Poisson likelihood node.

Usage

```
Poisson(input_dim = NULL)
```

Arguments

`input_dim`	a vector of length one that contains the indices of one GP node in the feeding layer whose outputs feed into this likelihood node. When set to `NULL`, all outputs from GP nodes in the feeding layer feed into this likelihood node, and in such a case one needs to ensure that only one GP node is specified in the feeding layer. Defaults to `NULL`.

Details

See further examples and tutorials at https://mingdeyu.github.io/dgpsi-R/.

Value

A 'python' object to represent a Poisson likelihood node.

Note

The Poisson likelihood node can only be linked to one feeding GP node.

Examples

```
## Not run:

# Check https://mingdeyu.github.io/dgpsi-R/ for examples
# on how to customize DGP structures using Poisson().

## End(Not run)
```

---

predict *Predictions from GP, DGP, or linked (D)GP emulators*

Description

This function implements single-core or multi-core predictions (with or without multi-threading) from GP, DGP, or linked (D)GP emulators.

Usage

```
## S3 method for class 'dgp'
predict(
  object,
  x,
  method = "mean_var",
  full_layer = FALSE,
  sample_size = 50,
  cores = 1,
  chunks = NULL,
  threading = FALSE,
  ...
)

## S3 method for class 'lgp'
predict(
  object,
  x,
  method = "mean_var",
  full_layer = FALSE,
  sample_size = 50,
  cores = 1,
  chunks = NULL,
  threading = FALSE,
  ...
)

## S3 method for class 'gp'
predict(
  object,
  x,
  method = "mean_var",
  sample_size = 50,
  cores = 1,
  chunks = NULL,
  ...
)
```

Arguments

`object`	an instance of the `gp`, `dgp`, or `lgp` class.
`x`	the testing input data:

- if `object` is an instance of the `gp` or `dgp` class, `x` is a matrix where each row is an input testing data point and each column is an input dimension.
- if `object` is an instance of the `lgp` class, `x` can be a matrix or a list:
  - if `x` is a matrix, it is the global testing input data that feed into the emulators in the first layer of a system. The rows of `x` represent different input data points and the columns represent input dimensions across all emulators in the first layer of the system. In this case, it is assumed that the only global input to the system is the input to the emulators in the first layer and there is no global input to emulators in other layers.
  - if `x` is a list, it should have *L* (the number of layers in an emulator system) elements. The first element is a matrix that represents the global testing input data that feed into the emulators in the first layer of the system. The remaining *L-1* elements are *L-1* sub-lists, each of which contains a number (the same number of emulators in the corresponding layer) of matrices (rows being testing input data points and columns being input dimensions) that represent the global testing input data to the emulators in the corresponding layer. The matrices must be placed

in the sub-lists based on how their corresponding emulators are placed in `struc` argument of `lgp()`. If there is no global input data to a certain emulator, set `NULL` in the corresponding sub-list of `x`.

`method`	the prediction approach: mean-variance (`"mean_var"`) or sampling (`"sampling"`) approach. Defaults to `"mean_var"`.
`full_layer`	a bool indicating whether to output the predictions of all layers. Defaults to `FALSE`. Only used when `object` is a DGP and linked (D)GP emulator.
`sample_size`	the number of samples to draw for each given imputation if `method = "sampling"`. Defaults to `50`.
`cores`	the number of cores/workers to be used. If set to `NULL`, the number of cores is set to `(max physical cores available - 1)`. Defaults to `1`.
`chunks`	the number of chunks that the testing input matrix `x` will be divided into for multi-cores to work on. Only used when `cores` is not `1`. If not specified (i.e., `chunks = NULL`), the number of chunks is set to the value of `cores`. Defaults to `NULL`.
`threading`	a bool indicating whether to use the multi-threading to accelerate the predictions of DGP or linked (D)GP emulators. Turn this option on when you have a moderately large number of training data points as in such a case you could gain faster predictions. Defaults to `FALSE`.
`...`	N/A.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

- If `object` is an instance of the `gp` class:
  1. if `method = "mean_var"`: an updated `object` is returned with an additional slot called `results` that contains two matrices named `mean` for the predictive means and `var` for the predictive variances. Each matrix has only one column with its rows corresponding to testing positions (i.e., rows of `x`).
  2. if `method = "sampling"`: an updated `object` is returned with an additional slot called `results` that contains a matrix whose rows correspond to testing positions and columns correspond to `sample_size` number of samples drawn from the predictive distribution of GP.
- If `object` is an instance of the `dgp` class:
  1. if `method = "mean_var"` and `full_layer = FALSE`: an updated `object` is returned with an additional slot called `results` that contains two matrices named `mean` for the predictive means and `var` for the predictive variances respectively. Each matrix has its rows corresponding to testing positions and columns corresponding to DGP global output dimensions (i.e., the number of GP/likelihood nodes in the final layer).
  2. if `method = "mean_var"` and `full_layer = TRUE`: an updated `object` is returned with an additional slot called `results` that contains two sub-lists named `mean` for the predictive means and `var` for the predictive variances respectively. Each sub-list contains L (i.e., the number of layers) matrices named `layer1, layer2,..., layerL`. Each matrix has its

rows corresponding to testing positions and columns corresponding to output dimensions (i.e., the number of GP/likelihood nodes from the associated layer).

3. if `method = "sampling"` and `full_layer = FALSE`: an updated `object` is returned with an additional slot called `results` that contains *D* (i.e., the number of GP/likelihood nodes in the final layer) matrices named `output1, output2,..., outputD`. Each matrix has its rows corresponding to testing positions and columns corresponding to samples of size: `B * sample_size`, where `B` is the number of imputations specified in `dgp()`.
4. if `method = "sampling"` and `full_layer = TRUE`: an updated `object` is returned with an additional slot called `results` that contains *L* (i.e., the number of layers) sub-lists named `layer1, layer2,..., layerL`. Each sub-list represents samples drawn from the GP/likelihood nodes in the corresponding layer, and contains *D* (i.e., the number of GP/likelihood nodes in the corresponding layer) matrices named `output1, output2,..., outputD`. Each matrix gives samples of the output from one of *D* GP/likelihood nodes, and has its rows corresponding to testing positions and columns corresponding to samples of size: `B * sample_size`, where `B` is the number of imputations specified in `dgp()`.

- If `object` is an instance of the `lgp` class:

1. if `method = "mean_var"` and `full_layer = FALSE`: an updated `object` is returned with an additional slot called `results` that contains two sub-lists named `mean` for the predictive means and `var` for the predictive variances respectively. Each sub-list contains *M* number (same number of emulators in the final layer of the system) of matrices named `emulator1, emulator2,..., emulatorM`. Each matrix has its rows corresponding to global testing positions and columns corresponding to output dimensions of the associated emulator in the final layer.
2. if `method = "mean_var"` and `full_layer = TRUE`: an updated `object` is returned with an additional slot called `results` that contains two sub-lists named `mean` for the predictive means and `var` for the predictive variances respectively. Each sub-list contains *L* (i.e., the number of layers in the emulated system) components named `layer1, layer2,..., layerL`. Each component represents a layer and contains *M* number (same number of emulators in the corresponding layer of the system) of matrices named `emulator1, emulator2,..., emulatorM`. Each matrix has its rows corresponding to global testing positions and columns corresponding to output dimensions of the associated GP/DGP emulator in the corresponding layer.
3. if `method = "sampling"` and `full_layer = FALSE`: an updated `object` is returned with an additional slot called `results` that contains *M* number (same number of emulators in the final layer of the system) of sub-lists named `emulator1, emulator2,..., emulatorM`. Each sub-list corresponds to an emulator in the final layer, and contains *D* matrices, named `output1, output2,..., outputD`, that correspond to the output dimensions of the GP/DGP emulator. Each matrix has its rows corresponding to testing positions and columns corresponding to samples of size: `B * sample_size`, where `B` is the number of imputations specified in `lgp()`.
4. if `method = "sampling"` and `full_layer = TRUE`: an updated `object` is returned with an additional slot called `results` that contains *L* (i.e., the number of layers of the emulated system) sub-lists named `layer1, layer2,..., layerL`. Each sub-list represents a layer and contains *M* number (same number of emulators in the corresponding layer of the system) of components named `emulator1, emulator2,..., emulatorM`. Each component corresponds to an emulator in the associated layer, and contains *D* matrices, named `output1, output2,..., outputD`, that correspond to the output dimensions of

the GP/DGP emulator. Each matrix has its rows corresponding to testing positions and columns corresponding to samples of size: `B * sample_size`, where `B` is the number of imputations specified in `lgp()`.

Note

Any R vector detected in `x` will be treated as a column vector and automatically converted into a single-column R matrix.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

`read` *Load the stored emulator*

Description

This function loads the `.pkl` file that stores the emulator.

Usage

```
read(pkl_file)
```

Arguments

`pkl_file`	the path to and the name of the `.pkl` file where the emulator is stored.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

A GP, DGP or linked (D)GP emulator S3 class.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

set_linked_idx *Set linked indices*

Description

This function sets the linked indices of a GP or DGP emulator if the information is not provided when the emulator is constructed by `gp()` or `dgp()`.

Usage

```
set_linked_idx(object, idx)
```

Arguments

`object`	an instance of the S3 class `gp` or `dgp`.
`idx`	indices of columns in the pooled output matrix (formed by column-combining outputs of all emulators in the feeding layer) that will feed into the GP or DGP emulator represented by `object`.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

An updated `object` with the information of `idx` incorporated.

Note

This function is useful when different models are emulated by different teams. Each team can create their (D)GP emulator even without knowing how different emulators are connected together. When this information is available and different emulators are collected, the connection information between emulators can then be assigned to individual emulators with this function.

Examples

```
## Not run:

# See lgp() for an example.

## End(Not run)
```

`summary` *Summary of a constructed GP, DGP, or linked (D)GP emulator*

Description

This function summarizes key information of a GP, DGP or linked (D)GP emulator.

Usage

```
## S3 method for class 'gp'
summary(object, ...)

## S3 method for class 'dgp'
summary(object, ...)

## S3 method for class 'lgp'
summary(object, ...)
```

Arguments

`object`	can be one of the following: • the S3 class `gp`. • the S3 class `dgp`. • the S3 class `lgp`.
`...`	N/A.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

A table summarizing key information contained in `object`.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

trace_plot *Plot of DGP model parameter traces*

Description

This function plots the traces of model parameters of a chosen GP node in a DGP emulator.

Usage

```
trace_plot(object, layer = NULL, node = 1)
```

Arguments

`object`	an instance of the `dgp` class.
`layer`	the index of a layer. Defaults to `NULL` for the final layer.
`node`	the index of a GP node in the layer specified by `layer`. Defaults to `1` for the first GP node in the corresponding layer.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

A `ggplot` object.

Examples

```
## Not run:

# See dgp() for an example.

## End(Not run)
```

validate *Validate a constructed GP, DGP, or linked (D)GP emulator*

Description

This function validate a constructed GP, DGP, or linked (D)GP emulator via the Leave-One-Out (LOO) cross validation or Out-Of-Sample (OOS) validation.

Usage

```
validate(object, x_test, y_test, method, verb, force, cores, ...)

## S3 method for class 'gp'
validate(
  object,
  x_test = NULL,
  y_test = NULL,
  method = "mean_var",
  verb = TRUE,
  force = FALSE,
  cores = 1,
  ...
)

## S3 method for class 'dgp'
validate(
  object,
  x_test = NULL,
  y_test = NULL,
  method = "mean_var",
  verb = TRUE,
  force = FALSE,
  cores = 1,
  threading = FALSE,
  ...
)

## S3 method for class 'lgp'
validate(
  object,
  x_test = NULL,
  y_test = NULL,
  method = "mean_var",
  verb = TRUE,
  force = FALSE,
  cores = 1,
  threading = FALSE,
  ...
)
```

Arguments

`object` can be one of the following:

- the S3 class gp.
- the S3 class dgp.
- the S3 class `lgp`.

`x_test` the OOS testing input data:

- if `x` is an instance of the `gp` or `dgp` class, `x_test` is a matrix where each row is an input testing data point and each column is an input dimension.
- if `x` is an instance of the `lgp` class, `x_test` can be a matrix or a list:
  - if `x_test` is a matrix, it is the global testing input data that feed into the emulators in the first layer of a system. The rows of `x_test` represent different input data points and the columns represent input dimensions across all emulators in the first layer of the system. In this case, it is assumed that the only global input to the system is the input to the emulators in the first layer and there is no global input to emulators in other layers.
  - if `x_test` is a list, it should have *L* (the number of layers in an emulator system) elements. The first element is a matrix that represents the global testing input data that feed into the emulators in the first layer of the system. The remaining *L-1* elements are *L-1* sub-lists, each of which contains a number (the same number of emulators in the corresponding layer) of matrices (rows being testing input data points and columns being input dimensions) that represent the global testing input data to the emulators in the corresponding layer. The matrices must be placed in the sub-lists based on how their corresponding emulators are placed in `struc` argument of `lgp()`. If there is no global input data to a certain emulator, set `NULL` in the corresponding sub-list of `x_test`.

`x_test` must be provided for the validation if `x` is an instance of the `lgp`. Defaults to `NULL`.

`y_test` the OOS testing output data that correspond to `x_test`:

- if `x` is an instance of the `gp` class, `y_test` is a matrix with only one column and each row being an testing output data point.
- if `x` is an instance of the `dgp` class, `y_test` is a matrix with its rows being testing output data points and columns being output dimensions.
- if `x` is an instance of the `lgp` class, `y_test` can be a single matrix or a list of matrices:
  - if `y_test` is a single matrix, then there is only one emulator in the final layer of the linked emulator system and `y_test` represents the emulator's output with rows being testing positions and columns being output dimensions.
  - if `y_test` is a list, then `y_test` should have *M* number (the same number of emulators in the final layer of the system) of matrices. Each matrix has its rows corresponding to testing positions and columns corresponding to output dimensions of the associated emulator in the final layer.

`y_test` must be provided for the validation if `x` is an instance of the `lgp`. Defaults to `NULL`.

`method` the prediction approach in validations: mean-variance (`"mean_var"`) or sampling (`"sampling"`) approach. Defaults to `"mean_var"`.

`verb` a bool indicating if the trace information on validations will be printed during the function execution. Defaults to `TRUE`.

`force`	a bool indicating whether to force the LOO or OOS re-evaluation when `loo` or `oos` slot already exists in `object`. When `force = FALSE`, `validate()` will try to determine automatically if the LOO or OOS re-evaluation is needed. Set `force` to `TRUE` when LOO or OOS re-evaluation is required. Defaults to `FALSE`.
`cores`	the number of cores/workers to be used for the LOO or OOS validation. If set to `NULL`, the number of cores is set to `(max physical cores available - 1)`. Defaults to `1`.
`...`	N/A.
`threading`	a bool indicating whether to use the multi-threading to accelerate the LOO or OOS. Turning this option on could improve the speed of validations when the emulator is built with a moderately large number of training data points.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

- If `object` is an instance of the `gp` class, an updated `object` is returned with an additional slot called `loo` (for LOO cross validation) or `oos` (for OOS validation) that contains:
  - two slots called `x_train` (or `x_test`) and `y_train` (or `y_test`) that contain the validation data points for LOO (or OOS).
  - a column matrix called `mean`, if `method = "mean_var"`, or `median`, if `method = "sampling"`, that contains the predictive means or medians of the GP emulator at validation positions.
  - three column matrices called `std`, `lower`, and `upper` that contain the predictive standard deviations and credible intervals of the GP emulator at validation positions. If `method = "mean_var"`, the upper and lower bounds of a credible interval are two standard deviations above and below the predictive mean. If `method = "sampling"`, the upper and lower bounds of a credible interval are 2.5th and 97.5th percentiles.
  - a numeric value called `nrmse` that contains the (min-max) normalized root mean/median squared error of the GP emulator. The min-max normalization is based on the maximum and minimum values of the validation outputs contained in `y_train` (or `y_test`).

The rows of matrices (`mean`, `median`, `std`, `lower`, and `upper`) correspond to the validation positions.

- If `object` is an instance of the `dgp` class, an updated `object` is returned with an additional slot called `loo` (for LOO cross validation) or `oos` (for OOS validation) that contains:
  - two slots called `x_train` (or `x_test`) and `y_train` (or `y_test`) that contain the validation data points for LOO (or OOS).
  - a matrix called `mean`, if `method = "mean_var"`, or `median`, if `method = "sampling"`, that contains the predictive means or medians of the DGP emulator at validation positions.
  - three matrices called `std`, `lower`, and `upper` that contain the predictive standard deviations and credible intervals of the DGP emulator at validation positions. If `method = "mean_var"`, the upper and lower bounds of a credible interval are two standard deviations above and below the predictive mean. If `method = "sampling"`, the upper and lower bounds of a credible interval are 2.5th and 97.5th percentiles.

  - a vector called `nrmse` that contains the (min-max) normalized root mean/median squared errors of the DGP emulator across different output dimensions. The min-max normalization is based on the maximum and minimum values of the validation outputs contained in `y_train` (or `y_test`).

The rows and columns of matrices (`mean`, `median`, `std`, `lower`, and `upper`) correspond to the validation positions and DGP emulator output dimensions, respectively.

- If `object` is an instance of the `lgp` class, an updated `object` is returned with an additional slot called `oos` (for OOS validation) that contains:
  - two slots called `x_test` and `y_test` that contain the validation data points for OOS.
  - a list called `mean`, if `method = "mean_var"`, or `median`, if `method = "sampling"`, that contains the predictive means or medians of the linked (D)GP emulator at validation positions.
  - three lists called `std`, `lower`, and `upper` that contain the predictive standard deviations and credible intervals of the linked (D)GP emulator at validation positions. If `method = "mean_var"`, the upper and lower bounds of a credible interval are two standard deviations above and below the predictive mean. If `method = "sampling"`, the upper and lower bounds of a credible interval are 2.5th and 97.5th percentiles.
  - a list called `nrmse` that contains the (min-max) normalized root mean/median squared errors of the linked (D)GP emulator. The min-max normalization is based on the maximum and minimum values of the validation outputs contained in `y_test`.

Each element in `mean`, `median`, `std`, `lower`, `upper`, and `nrmse` corresponds to a (D)GP emulator in the final layer of the linked (D)GP emulator.

Note

- When both `x_test` and `y_test` are `NULL`, the LOO cross validation will be implemented. Otherwise, OOS validation will be implemented. The LOO validation is only applicable to a GP or DGP emulator (i.e., `x` is an instance of the `gp` or `dgp` class). If a linked (D)GP emulator (i.e., `x` is an instance of the `lgp` class) is provided, `x_test` and `y_test` must also be provided for OOS validation.
- Any R vector detected in `x_test` and `y_test` will be treated as a column vector and automatically converted into a single-column R matrix.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

`write` *Save the constructed emulator*

Description

This function saves the constructed emulator to a `.pkl` file.

Usage

```
write(object, pkl_file)
```

Arguments

`object`	an instance of the S3 class `gp`, `dgp`, or `lgp`.
`pkl_file`	the path to and the name of the `.pkl` file to which the emulator `object` is saved.

Details

See further examples and tutorials at `https://mingdeyu.github.io/dgpsi-R/`.

Value

No return value. `object` will be save to a local `.pkl` file specified by `pkl_file`.

Note

Since the constructed emulators are 'python' objects, `save()` from R will not work as it is only for R objects.

Examples

```
## Not run:

# See gp(), dgp(), or lgp() for an example.

## End(Not run)
```

Index