Package 'MaxWiK'

November 4, 2022

Title Machine learning method based on Maxima Weighted Isolation Kernel mean embedding

Version 1.0.0

Description This software is a package named **MaxWiK** contains Approximate Bayesian Computation method to choose a single parameter for a single observation point. The method involves the transformation of row data to a Hilbert space (mapping) and the measurement of the similarity between simulated points and maxima weighted Isolation Kernel mapping related to the observation point. We also design a heuristic algorithm for parameter estimation that requires no calculation and is dimension independent.

License AGPL (>= 3)

Depends R (>= 3.6.0)

Imports graphics,
grDevices,
randomcoloR,
methods,
stats,
utils,
scales,
bayestestR,
parallel,
plotly,
kernlab,
abc,
magrittr,
ggplot2

Suggests rmarkdown,
knitr,
testthat,
DiagrammeR

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

VignetteBuilder knitr

LazyData true

R topics documented:

adjust_Gram 2

analyze_experiments . . . 3
check_numeric_format . . . 4
check_packages . . . 4
check_pkg . . . 5
copy_pipelines . . . 5
experiment_models . . . 6
Gaussian_model . . . 7
gen_colors . . . 8
get_inverse_GRAM . . . 9
get_kernel_mean_embedding . . . 10
Get_MAP . . . 10
Get_parameter . . . 11
get_par_best_from_simnet . . . 12
GET_SUBSET . . . 13
get_subset_of_feature_map . . . 13
get_voronoi_feature . . . 14
get_voronoi_feature_PART_dataset . . . 15
iKernel . . . 15
K2_ABC . . . 17
Mean_iKernel_parameters . . . 18
model . . . 19
MSE_sim . . . 20
norm_vec . . . 21
plot_2D . . . 21
plot_2D_lines . . . 22
plot_sudoku_2D . . . 24
read_file . . . 26
restrict_data . . . 26
Results_toy_experiments . . . 32
sampler_method . . . 33
simulation_example . . . 36
sudoku . . . 38

Index **41**

`adjust_Gram` *Function to get Gram matrix after adjusting of sigma parameter of a kernel*

Description

Function to get Gram matrix after adjusting of sigma parameter of a kernel

Usage

```
adjust_Gram(kernel, sigma = (2^(1:20)) * 0.001, x, y)
```

Arguments

kernel	Function of kernel with parameter sigma, class from kernel lab package
sigma	numeric vector of possible values of the sigma parameter for a kernel function, by default sigma = (2**(1:20)) * 1E-3
x	Matrix of stat.sim
y	Matrix of stat.obs

Value

adjust_Gram function returns Gram matrix after adjusting of sigma

Examples

```
NULL
```

---

analyze_experiments *Function to get statistics from the results of experiments and find the best methods for each simulation case*

Description

Function to get statistics from the results of experiments and find the best methods for each simulation case

Usage

```
analyze_experiments(DF, file_to_save = "../gplot.pdf")
```

Arguments

DF	Data frame with results of experiments, output of experiment_models() function.

Value

List of statistical data with analysis of results:

- ;
- ;
- ;
- ;
- ;
- ;
- .

Examples

```
NULL
```

`check_numeric_format` *Function to check DATA.FRAME*

Description

Check that DATA.FRAME has numeric format for ALL the columns and it has NO 'NA' values

Usage

```
check_numeric_format(l)
```

Arguments

l	DATA.FRAME that should have data of numeric type

Value

TRUE if data.frame has ONLY numeric data and FALSE vice verse

Examples

```
## Not run:
check_numeric_format( data.frame( A= c(9,0), B = c(4,6)) )  # TRUE
check_numeric_format( data.frame( A= c(9,0), B = c(4,NA)) )  # Error due to NA value
check_numeric_format( data.frame( A= c(9,'0'), B = c(4,6)) ) # Error due to character in the data

## End(Not run)
```

`check_packages` *Check the installation of packages and attach them with corresponding functions*

Description

Check the installation of packages and attach them with corresponding functions

Usage

```
check_packages(pkgs = NULL)
```

Arguments

pkgs	List of package names with related function names, by default (or when pkgs = NULL) the list of packages are described in Namespace file of the package or 'R/MaxWiK-package.R' file

Value

if the packages are installed then it returns NULL else it returns error message

Examples

```
check_packages(  )
```

---

check_pkg *Check the installation of a package for some functions*

Description

Check the installation of a package for some functions

Usage

```
check_pkg(pkg)
```

Arguments

pkg	Package name

Value

if the package is installed then it returns NULL else it returns error message

Examples

```
check_pkg( pkg = 'grDevices' )
```

---

copy_pipelines *Function to copy the pipelines from extdata folder in the library to /Pipelines/ folder in the working directory*

Description

Function to copy the pipelines from extdata folder in the library to /Pipelines/ folder in the working directory

Usage

```
copy_pipelines(dir = "./")
```

Arguments

dir	Folder to where files should be save, by default dir = './'

Value

List of logic numbers for each copied file, TRUE - success, FALSE - not success

Examples

```
copy_pipelines( dir = 'Input' )
```

`experiment_models` *Function to prepare toy experiments*

Description

Function to prepare toy experiments

Usage

```
experiment_models(
  file_name = "output.txt",
  models = c("Gaussian", "Linear"),
  dimensions = (1:20) * 2,
  stochastic_terms = c(0, 0.1, 0.3, 0.7, 1, 1.5),
  rng = c(0, 10),
  restrict_points_number = 300
)
```

Arguments

`file_name`	Name of file to output results
`models`	Names of models for simulation, by default `models = c( 'Gaussian', 'Linear' )`
`dimensions`	Dimensions of models, by default `dimensions = (1:20)*2`
`stochastic_terms`	Stochastic terms for each model, by default `stochastic_terms = c( 0, 0.1, 0.3, 0.7, 1 )`
`rng`	Range for each variable, by default `rng = c( 0,10 )`

Value

`experiment_models()` returns list of results of experiments

Examples

```
NULL
```

`Gaussian_model` *The model of simulations that is based on Gaussian functions for each dimension*

Description

The function `Gaussian_model()` allows to generate parameters and statistics of simulations that are based on Gaussian function for each dimension:
`f( x ) = { exp( - (x1-x01)^2 / 2 ), ..., exp( - (xn-x0n)^2 / 2 ) } = { y1, y2, ..., yn }` is a vector of output data

The function `linear_model()` allows to generate parameters and statistics of simulations that are based on linear function for each dimension:
`f( x ) = { 1 + ( x1 - x01 ) / x01 + noise_1, ..., 1 + ( xn - x0n ) / x0n + noise_n } = { y1, y2, ..., yn }` is a vector of output data

Usage

```
Gaussian_model(
  d = 1,
  x0 = 3,
  probability = TRUE,
  n = 1000,
  r = range(0, 10),
  noise = 0
)

linear_model(
  d = 1,
  x0 = 3,
  probability = TRUE,
  noise = 0.2,
  n = 1000,
  r = range(0, 10)
)
```

Arguments

`d`	Dimension of the parameter and model space
`x0`	Numeric vector with length of dimensionality of data frame, that contents the truth value of parameter. Each number in the vector should be within the range r
`probability`	Logical, if TRUE then apply uneven distribution for parameters generation
`n`	Integer number of points in data frames
`r`	Range `r = c(min, max)`, by default `r = range(0,10)`
`noise`	Noise factor, implemented as coefficient in `f(x) = ... + noise * runif(1)`

Value

The function `Gaussian_model()` returns list of two objects:

- stat.sim - data frame of simulations statistics,
- par.sim - data frame of parameters,
- stat.obs - data frame of an observation point.

The function `linear_model()` returns list of two objects:

- stat.sim - data frame of simulations statistics,
- par.sim - data frame of parameters,
- stat.obs - data frame of an observation point.

Functions

- `linear_model()`: The model of simulations that is based on linear functions for each dimension

Examples

```
NULL
NULL
```

---

gen_colors *Function to make a large number of colors*

Description

Function to make a large number of colors

Usage

```
gen_colors(nm = 12)
```

Arguments

`nm`	Number of colors

Value

Vector of colors with length more than nm

Examples

```
clrs = gen_colors( nm = 120 )
```

get_inverse_GRAM *The function to get inverse Gram matrix*

Description

Function get_inverse_GRAM() allows to get inverse Gram matrix based on given positive regularization constant lambda

Function check_positive_definite() returns logical value about n trials on 'is Gram matrix positive definite or not?' Just incorrect trial returns FALSE

Usage

```
get_inverse_GRAM(G, l = 1e-06, check_pos_def = FALSE)

check_positive_definite(G, n = 10)
```

Arguments

G	Gram matrix gotten via GRAM_iKernel() function
l	Lambda parameter or positive regularization constant
check_pos_def	Logical parameter to check the Gram matrix is positive definite or do not check
n	Number of iterations to check the positive definite property

Value

Function get_inverse_GRAM() returns the inverse Gram matrix based on the given positive regularization constant lambda l

Function check_positive_definite() returns logical value:
TRUE if Gram matrix is positive definite, and FALSE if it is not

Functions

- check_positive_definite(): The function to check the positive definite property of Gram matrix

Examples

```
NULL
NULL
```

get_kernel_mean_embedding *The function to calculate Maxima weighted kernel mean mapping for Isolation Kernel in RKHS related to parameters space*

Description

The function to calculate Maxima weighted kernel mean mapping for Isolation Kernel in RKHS related to parameters space

Usage

```
get_kernel_mean_embedding(parameters_Matrix_iKernel, Hilbert_weights)
```

Arguments

parameters_Matrix_iKernel	Matrix of all the points represented in RKHS related to parameters space
Hilbert_weights	Maximal weights in RKHS to get related part of kernel mean embedding from parameters_Matrix_iKernel

Value

Maxima weighted kernel mean mapping in the form of integer vector with length t (number of trees). Each element of the vector is index of Voronoi cell with maximal weight in the Voronoi diagram

Examples

```
NULL
```

Get_MAP *The function to get Maximum A Posteriori from numeric data frame*

Description

The function Get_MAP returns the Maximum A Posteriori (MAP) of data frame.

Usage

```
Get_MAP(DF)
```

Arguments

DF	Data frame of the integer or numeric numbers or characters

Value

The function Get_MAP returns the MAP for each dimension in data frame DF of vector

Examples

```
NULL
```

`Get_parameter` *Function to call a method to get parameter estimation and MSE for each used method*

Description

Function to call a method to get parameter estimation and MSE for each used method

Usage

```
Get_parameter(
  method_name,
  kernel_name = "",
  stat.obs,
  stat.sim,
  par.sim,
  G = NULL,
  par.truth
)
```

Arguments

`method_name`	Name of a method
`kernel_name`	Name of kernel function
`stat.obs`	Data frame of statistics of observation point
`stat.sim`	Data frame of statistics of simulations
`par.sim`	Data frame of parameters
`G`	Matrix of similarities for K2-ABC based on isolation kernel
`par.truth`	Truth parameter value to check result of estimation

Value

`par.truth,()` returns the list:
method_name = method_name,

- kernel_name,
- model_name,
- stochastic_term,
- MSE,
- running_time,
- iteration.

Examples

```
NULL
```

`get_par_best_from_simnet` *Function to extract all the best parameters estimation from results of the function* `simulation_example_many_psi_t()`

Description

Function to extract all the best parameters estimation from results of the function `simulation_example_many_psi_t()`

Usage

```
get_par_best_from_simnet(simnet)

get_spiderweb_from_simnet(simnet)

get_network_from_simnet(simnet)
```

Arguments

`simnet`	results of the function `simulation_example_many_psi_t()`

Value

`get_par_best_from_simnet()` returns all the best parameters estimation

`get_spiderweb_from_simnet()` returns all the spiderwebs from results of the function `simulation_example_many_ps`

`get_network_from_simnet()` returns all the networks from results of the function `simulation_example_many_psi_t(`

Functions

- `get_spiderweb_from_simnet()`: Function to extract all the spiderwebs from results of the function `simulation_example_many_psi_t()`
- `get_network_from_simnet()`: Function to extract all the networks from results of the function `simulation_example_many_psi_t()`

Examples

```
NULL
NULL
NULL
```

GET_SUBSET *The function to get subset with size psi for Voronoi diagram*

Description

The function to get subset with size psi for Voronoi diagram

Usage

```
GET_SUBSET(data_set, pnts)
```

Arguments

data_set	Data.frame of Voronoi diagram
pnts	Integer vector of indexes of columns of the data_set

Value

Subset of data_set with columns pnts

Examples

```
NULL
```

get_subset_of_feature_map *The function to get subset of points based on feature mapping*

Description

The function to get subset of points based on feature mapping

Usage

```
get_subset_of_feature_map(dtst, Matrix_Voronoi, iFeature_point)
```

Arguments

dtst	Dataset of all the original points
Matrix_Voronoi	Matrix of Voronoi diagrams based on the Isolation Kernel algorithm
iFeature_point	Feature mapping in RKHS for a point, that can be gotten via add_new_point_iKernel() function

Value

The subset of dtst that has points extracted with feature mapping of an observation point (iFeature_point)

Examples

```
NULL
```

`get_voronoi_feature` *The function to get feature representation in RKHS based on Voronoi diagram for WHOLE dataset*

Description

The function to get feature representation in RKHS based on Voronoi diagram for WHOLE dataset

Usage

```
get_voronoi_feature(
  psi = 40,
  t = 350,
  data,
  talkative = FALSE,
  new = TRUE,
  Matrix_Voronoi = NULL
)

add_new_point_iKernel(data, d1, Matrix_Voronoi, dissim, t, psi, nr)
```

Arguments

`psi`	Integer number related to the size of each Voronoi diagram
`t`	Integer number of trees in Isolation Kernel or dimension of RKHS
`data`	dataset of points, rows - points, columns - dimensions of a point
`talkative`	logical. If TRUE then print messages, FALSE for the silent execution
`new`	logical. Is Matrix_Voronoi new ? If TRUE then function will calculate Matrix_Voronoi, if FALSE function will use input Matrix_Voronoi.
`Matrix_Voronoi`	Matrix of Voronoi diagrams that is used only if new = FALSE
`d1`	Data point - usually it is an observation data point
`dissim`	Matrix of dissimilarity or distances between all points.
`nr`	Integer number of rows in matrix of distances (dissim) and also the size of dataset

Value

Feature representation in RKHS based on Voronoi diagram for WHOLE dataset

RKHS mapping for a new point based on Isolation Kernel mapping

Functions

- `add_new_point_iKernel()`: The function to get RKHS mapping based on Isolation Kernel for a new point

Examples

```
NULL
NULL
```

get_voronoi_feature_PART_dataset

The function to get feature representation in RKHS based on Voronoi diagram for PART of dataset

Description

`get_voronoi_feature_PART_dataset()` function returns the feature (mapping) representation in RKHS based on Voronoi diagram for NEW PART of dataset. The `Matrix_Voronoi` is based on the PREVIOUS dataset. The NEW PART of dataset will appear at the end of PREVIOUS dataset

Usage

```
get_voronoi_feature_PART_dataset(
  data,
  talkative = FALSE,
  start_row,
  Matrix_Voronoi
)
```

Arguments

`data`	Data.frame of new points
`talkative`	Logical parameter to print or do not print messages
`start_row`	Row number from which a new data should be added
`Matrix_Voronoi`	Matrix of Voronoi diagrams based on the PREVIOUS dataset

Value

List of three matrices: Matrix_Voronoi, Matrix_iKernel and dissim

Examples

```
NULL
```

iKernel

Function returns the value of similarity or Isolation KERNEL for TWO points

Description

`iKernel()` function returns value of similarity or Isolation KERNEL for TWO points that is number in the range `[0,1]`

`iKernel_point_dataset()` function returns vector of values of similarity based on Isolation Kernel between a new point and all the points of dataset

`get_weights_iKernel()` function returns list of two objects: the first object is numeric vector of weights for RKHS space, and the second object is numeric vector of weights of similarity for iFeature_point corresponding observation point

`GRAM_iKernel()` is the function to calculate Gram matrix for Isolation Kernel method based on Voronoi diagrams

Usage

```
iKernel(Matrix_iKernel, pnt_1, pnt_2, t)

iKernel_point_dataset(Matrix_iKernel, t, nr, iFeature_point)

get_weights_iKernel(GI, Matrix_iKernel, t, nr, iFeature_point)

GRAM_iKernel(Matrix_iKernel, check_pos_def = FALSE)
```

Arguments

Matrix_iKernel	Matrix of indexes of Voronoi cells for each point and each tree based on Isolation Kernel calculation
pnt_1	The first point of dataset
pnt_2	The second point of dataset
t	is a number of columns of Matrix_iKernel or dimension of Matrix_iKernel (corresponding to the number of trees t)
nr	is number of rows in Matrix_iKernel or size of dataset
iFeature_point	Feature mapping in RKHS for a new point, that can be gotten via add_new_point_iKernel() function
GI	The inverse Gram matrix
check_pos_def	Logical parameter to check the Gram matrix is positive definite or do not check

Value

The function iKernel() returns a value of similarity or Isolation KERNEL for TWO points

The function iKernel_point_dataset() returns a value of Isolation Kernel between a new point and dataset represented via Matrix_iKernel

The function get_weights_iKernel() returns the list of weights for RKHS space and weights of similarity for iFeature_point

The function GRAM_iKernel() returns Gram matrix of Isolation Kernel

Functions

- iKernel_point_dataset(): The function to get Isolation Kernel between a new point and dataset
- get_weights_iKernel(): The function to get weights from Feature mapping
- GRAM_iKernel(): The function to calculate Gram matrix for Isolation Kernel method

Examples

```
NULL
NULL
NULL
NULL
```

K2_ABC *Function to get parameter estimation and weights using K2-ABC method*

Description

`K2_ABC()` function allows to get parameter estimation and weights using K2-ABC method described in the paper Mijung Park, Wittawat Jitkrittum, Dino Sejdinovic, Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, PMLR 51:398-407, 2016.

`adjust_K2_ABC()` allows to adjust epsilon parameter for K2-ABC method using numeric vector of epsilon, find parameter estimation for each epsilon and choose the best one.

`adjust_K2_ABC_iKernel()` allows to adjust epsilon parameter for K2-ABC method using numeric vector of epsilon, find parameter estimation for each epsilon and choose the best one based on matrix of isolation kernel.

`adjust_ABC_tolerance()` allows to adjust tolerance parameter for rejection ABC method using numeric vector of tolerance, find parameter estimation for each tolerance and choose the best one.

Usage

```
K2_ABC(G, epsilon = 0.5, par.sim)

adjust_K2_ABC(
  epsilon = c(0.01, 0.02, 0.03, 0.04, (0.05 * 1:20)),
  par.sim,
  stat.sim,
  stat.obs,
  kernel
)

adjust_K2_ABC_iKernel(
  epsilon = c(0.01, 0.02, 0.03, 0.04, (0.05 * 1:20)),
  par.sim,
  stat.sim,
  stat.obs,
  G
)

adjust_ABC_tolerance(
  tolerance = c(0.001, 0.002, 0.005, (0.01 * 1:20)),
  par.sim,
  stat.sim,
  stat.obs
)
```

Arguments

`G`	Kernel matrix `G` containers similarities between simulation points and observation point based on isolation kernel
`epsilon`	Numeric vector of possible values of epsilon, by default `epsilon = (0.05 * 1:20)`

par.sim	dataset/matrix of parameters for simulation
stat.sim	Matrix of statistics of simulations
stat.obs	Matrix of statistics of an observation
kernel	Kernel function of class kernel from kernlab package
tolerance	Vector of tolerance values for rejection ABC method to get the best one, by default tolerance = c(0.001, 0.002, 0.005, (0.01 * 1:20)

Value

K2_ABC() returns the list of:

1. weights for par.sim related to observation point based on Gram matrix
2. parameter estimation par.est

adjust_K2_ABC() returns the best parameter estimation using K2-ABC method varying epsilon

adjust_K2_ABC_iKernel() returns the best parameter estimation using K2-ABC method varying epsilon and based on isolation kernel

adjust_ABC_tolerance() returns the best parameter estimation using rejection ABC method varying tolerance and tolerance value

Functions

- adjust_K2_ABC(): Function to adjust epsilon parameter for K2-ABC method
- adjust_K2_ABC_iKernel(): Function to adjust epsilon parameter for K2-ABC method
- adjust_ABC_tolerance(): Function to adjust tolerance parameter for rejection ABC method

Examples

```
NULL
NULL
NULL
NULL
```

Mean_iKernel_parameters

The function returns the weighted mean of the parameter based on Isolation Kernel ABC method

---

Description

The function Mean_iKernel_parameters() returns the weighted mean of the parameter that was calculated with KernelABC() function that represents Isolation Kernel ABC method

Usage

```
Mean_iKernel_parameters(param, sm)
```

Arguments

param	Data frame of parameters
sm	Numeric vector of weights gotten from iKernelABC() function

Value

The function Mean_iKernel_parameters() returns the weighted mean of the parameter

Examples

```
NULL
```

---

model *Model definition for sampler to calculate one simulation*

Description

Model definition for sampler to calculate one simulation

Usage

```
model(name = c("Gaussian", "Linear")[1], parameter, x0, stat.obs, noise = 0)
```

Arguments

name	Name of a model, can be either 'Gaussian' or 'Linear'
parameter	Value of a parameter
x0	True value of parameter
stat.obs	Data frame of statistics of observation point
noise	Value of stochastic term

Value

model() returns data frame with a result of a simulation based on the model

Examples

```
NULL
```

MSE_sim *The function to get the mean square error values for statistics of simulations*

Description

The function MSE_sim() allows to get the mean square error values for statistics of simulations

The function MSE_parameters() allows to get MSE for parameters if the truth parameter is known

Usage

```
MSE_sim(stat.obs, stat.sim)

MSE_parameters(par.truth, par.top = NULL, par.best)
```

Arguments

stat.obs	Summary statistics of the observation point
stat.sim	Summary statistics of the simulations (model output)
par.truth	The truth parameter
par.top	Parameters from the top of similarities of iKernelABC() algorithm
par.best	The best parameter from iKernelABC() algorithm

Value

The function MSE_sim() returns numeric vector of the mean square error values for statistics of simulations

The function MSE_parameters() returns list of two numbers:

- mean of MSE values for all the points from par.top;
- MSE value for the point of par.best

Functions

- MSE_parameters(): The function calculates mean square error (MSE) value for parameters as differences between them and already the known truth parameter

Examples

```
NULL
NULL
```

norm_vec *The norm function for vector*

Description

The norm function for vector

Usage

```
norm_vec(x)

norm_vec_sq(x)
```

Arguments

x	numeric vector

Value

The squared root of sum of squared elements of the vector x or Euclid length of the vector x

The squared Euclid norm or the sum of squared elements of the vector x

Functions

- `norm_vec_sq()`: The squared norm or the sum of squared elements of the vector x

Examples

```
norm_vec(c(3,4)) # that returns 5
norm_vec(c(12,5)) # that returns 13

norm_vec_sq(c(3,4)) # that returns 25
norm_vec_sq(c(12,5)) # that returns 169
```

plot_2D *Function to plot 2D figure of points y = y(x)*

Description

Function to plot 2D figure of points y = y(x)

Usage

```
plot_2D(
  x,
  y,
  names = c("X", "Y"),
  pch = 18,
  col = "blue",
  cex = 1.2,
```

```
  xr = c(-10, 10),
  yr = c(-10, 10),
  safe_pdf = FALSE,
  filename = "./plot.pdf",
  change_par = TRUE
)
```

Arguments

x	Input data for axes X
y	Input data for axes Y
names	Vector of two characters with names for X and Y axes
pch	Parameter pch for plot function
col	Colors of points
cex	Parameter cex for plot function
xr	Range for X
yr	Range for Y
safe_pdf	Indicator to save plot to a file or not
filename	Name of file to save plot if safe_pdf == TRUE
change_par	Indicator to change par() or not for a plot. By default change_par = TRUE, after plot it will be returned to initial values

Value

NULL, making 2D plot using points

Examples

```
plot_2D( x=-5:5, y=-3:7 )
```

---

plot_2D_lines *Function to plot 2D figure of lines* `y_i = DF[, nl[ i ] ])` *, i - index*

Description

Function to plot 2D figure of lines `y_i = DF[, nl[ i ] ])` , i - index

Usage

```
plot_2D_lines(
  x,
  DF,
  nl = 1:2,
  names = c("X", "Y"),
  legend_names = "",
  col = c("blue3", "darkmagenta", "red", "green4", "darkorange", "steelblue1"),
  cex = 1.2,
  lwd = 2,
  lt = c(1:6),
```

```
  xr = c(-10, 10),
  yr = c(-10, 10),
  safe_pdf = FALSE,
  filename = "./plot.pdf",
  type = "l",
  logscale = "",
  draw_key = TRUE,
  change_par = TRUE
)
```

Arguments

x	Input data for axes X
DF	data.frame with data to plot
nl	indexes of columns in DF to plot
names	Vector of two characters with names for X and Y axes
legend_names	Name of legend
col	Vector of colors for lines
cex	Parameter cex for plot function
lwd	Vector of width of lines
lt	Vector of types of lines
xr	Range for X
yr	Range for Y
safe_pdf	Indicator to save plot to a file or not
filename	Name of file to save plot if safe_pdf == TRUE
type	Parameter type in plot function
logscale	Parameter logscale in plot function
draw_key	Indicator to draw key or not
change_par	Indicator to change par() or not for a plot. By default change_par = TRUE, after plot it will be returned to initial values

Value

NULL, making 2D plot using lines

Examples

```
NULL
# plot_2D_lines( x = DF[, 1 ], DF, nl = 8:12 , xr = c(1,max(DF$Time) ), yr = c(0,1) )
# xr = c(1,max(DF$Time) )
# yr = c(0,max(DF[,14],DF[,16],DF[,17] ))
# plot_2D_lines( x = DF[, 1 ], DF, nl = c(14,16,17) , xr =xr, yr = yr )
# plot_2D_lines( x = DF[, 1 ], DF, nl = 18:22 , xr = c(1,max(DF$Time) ), yr = c(0,1) )
```

plot_sudoku_2D *Function to plot all the results of the function* sudoku()

Description

The function plot_sudoku_2D draws all the results of the function sudoku()

The function plot_web_2D() draws all the results of the function spiderweb()

Usage

```
plot_sudoku_2D(
  stat.sim,
  par.sim,
  par.truth,
  iKernelABC,
  rslt,
  ind_X,
  ind_Y,
  names = c("Parameter_1", "Parameter_2"),
  xlim,
  ylim,
  show_tracer = TRUE,
  show_obs = TRUE,
  show_appropriate = TRUE,
  show_best = TRUE,
  show_u_point = TRUE,
  show_legend = FALSE
)

plot_web_2D(
  stat.sim,
  par.sim,
  par.truth,
  iKernelABC,
  web,
  ind_X,
  ind_Y,
  names = c("P1", "P2"),
  xlim,
  ylim,
  show_tracer = TRUE,
  show_obs = TRUE,
  show_network = TRUE,
  show_best = TRUE,
  show_u_point = TRUE,
  show_legend = FALSE
)
```

Arguments

stat.sim	Summary statistics of the simulations (model output)

par.sim	Data frame of parameters of the model
par.truth	Truth value of the parameter corresponding to observation point (if known)
iKernelABC	Result of calculations based on Isolation Kernel ABC that can be gotten by the function iKernelABC()
rslt	Results of function sudoku()
ind_X	Column index of the par.sim data frame to plot as X axes
ind_Y	Column index of the par.sim data frame to plot as Y axes
names	Vector of axes names, by default names = c('Parameter_1', 'Parameter_2')
xlim	Numeric vector of the range for X axes
ylim	Numeric vector of the range for Y axes
show_tracer	Logical to show or do not show all the tracer points
show_obs	Logical to show or do not show the stat.obs or parameter of observation point (if known)
show_appropriate	Logical to show or do not show all the points of the object rslt$surroundings_best_points from results of sudoku() function
show_best	Logical to show or do not show the best point of the results of sudoku() function
show_u_point	Logical to show or do not show the estimated point of parameter for an observation
show_legend	Logical to show or do not show the legend
web	Results of the function spiderweb() to draw on the plot
show_network	Logical to show or do not show network points

Value

Plot all the results of sudoku() function

The function plot_web_2D() draws all the results of the function spiderweb()

Functions

- plot_web_2D(): The function to plot all the results of the function spiderweb()

Examples

```
NULL
NULL
```

read_file *Function to read file*

Description

Function to read file

Usage

```
read_file(file_name = "", stringsAsFactors = FALSE, header = TRUE)
```

Arguments

file_name	Name of file to read
stringsAsFactors	Parameter for read.table function, by default stringsAsFactors = FALSE
header	Logical type to read or do not read head of a file

Value

data.frame of data from a file

Examples

```
# fl = system.file('extdata/Input', 'gene_map.txt',package = 'tugHall.3', mustWork = TRUE )
# read_file(file_name = fl, stringsAsFactors = FALSE )
# fl = system.file('extdata/Input', 'CF.txt',package = 'tugHall.3', mustWork = TRUE )
# read_file(file_name = fl, stringsAsFactors = FALSE, header = FALSE )
NULL
```

restrict_data *Function to get Approximate Bayesian Computation based on Maxima Weighted Isolation Kernel mapping*

Description

`restrict_data()` is based on rejection ABC method to restrict original dataset

The function `iKernelABC()` is used to get Approximate Bayesian Computation based on Maxima Weighted Isolation Kernel mapping. On given data frame of parameters, statistics of the simulations and an observation, using the internal parameters psi and t, the function `iKernelABC()` returns the estimation of a parameter corresponding to Maxima weighted Isolation Kernel ABC method.

The function `spiderweb()` itteratively generates tracer points gotten from `sudoku()` algorithm, based on the simple procedure:

- making a reflection of the top points from the best point,
- and then generating the point tracers between them,

- finally, the algorithm chooses again the top points and the best point (`sudoku()` function is used),
- repeat all the steps until condition to be `TRUE`:
  `abs( max( sim_tracers ) - sim_previous ) < epsilon`

The function `spiderweb()` itteratively generates tracer points gotten from `sudoku()` algorithm, based on the simple procedure:

- making a reflection of the top points from the best point,
- and then generating the point tracers between them,
- finally, the algorithm chooses again the top points and the best point (`sudoku()` function is used),
- repeat all the steps until condition to be `TRUE`:
  `abs( max( sim_tracers ) - sim_previous ) < epsilon`

The function `spiderweb_slow()` itteratively generates tracer points gotten from `sudoku()` algorithm, based on the simple procedure:

- making a reflection of the top points from the best point,
- and then generating the point tracers between them,
- finally, the algorithm chooses again the top points and the best point (`sudoku()` function is used),
- repeat all the steps until condition to be `TRUE`:
  `abs( min( sim_tracers ) - sim_previous ) < epsilon`

Usage

```
restrict_data(par.sim, stat.sim, stat.obs, size = 300)

sampler_MaxWiK(
  stat.obs,
  stat.sim,
  par.sim,
  model,
  arg0 = list(),
  size = 500,
  psi_t,
  epsilon,
  nmax = 100,
  include_top = FALSE,
  slowly = FALSE,
  rate = 0.2
)

iKernelABC(
  psi = 40,
```

```
  t = 350,
  param,
  stat.sim,
  stat.obs,
  talkative = FALSE,
  check_pos_def = TRUE
)

Get_iKernel_estimation(iKernelABC, par.sim, stat.sim, stat.obs)

adjust_psi_t(
  par.sim,
  stat.sim,
  stat.obs,
  talkative = FALSE,
  check_pos_def = FALSE,
  n_best = 10,
 psi_t = data.frame(psi = as.numeric(sapply(X = c(2:8) * 2, FUN = function(x) rep(x,
    8))), t = rep(c(4, 6, 8, 10, 12, 14, 16, 20), 7)),
  cores = 4
)

spiderweb_old(
  psi = 4,
  t = 35,
  param = param,
  stat.sim = stat.sim,
  stat.obs = stat.obs,
  talkative = FALSE,
  check_pos_def = FALSE,
  n_bullets = 5,
  n_best = 10,
  halfwidth = 0.5,
  epsilon = 0.001
)

spiderweb(
  psi = 4,
  t = 35,
  param = param,
  stat.sim = stat.sim,
  stat.obs = stat.obs,
  talkative = FALSE,
  check_pos_def = FALSE,
  n_bullets = 16,
  n_best = 10,
  halfwidth = 0.5,
  epsilon = 0.001,
  rate = 0.1,
  max_iteration = 5,
  save_web = TRUE
)
```

```
spiderweb_slow(
  psi = 4,
  t = 35,
  param = param,
  stat.sim = stat.sim,
  stat.obs = stat.obs,
  talkative = FALSE,
  check_pos_def = FALSE,
  n_bullets = 16,
  n_best = 10,
  halfwidth = 0.5,
  epsilon = 0.001,
  rate = 0.1,
  max_iteration = 15,
  save_web = TRUE
)
```

Arguments

par.sim	Data frame of parameters
stat.sim	Summary statistics of the simulations (model output)
stat.obs	Summary statistics of the observation point
size	Number of point to restrict original dataset
model	is a function to get output of simulation during sampling
arg0	is a list with arguments for a model function, so that arg0 is NOT changed during sampling
psi_t	Initial data.frame of psi and t, by default psi_t = data.frame(psi = as.numeric(sapply(X = c(2:8)*2, FUN = function(x) rep(x, 8))), t = rep(c(4,6,8,10,12,14,16,20), 7))
epsilon	Criterion to stop algorithm spiderweb() that isused to check: if (abs(max(sim_tracers) - sim_previous) < epsilon) break
nmax	is maximal number of iterations
include_top	Logical to include top points from spider_web() function to simulate or do not
slowly	Logical for two algorithms: slow and fast seekers in sampling
rate	Numeric rate from 0 to 1 that gives rate of changing of surround points of proposed max of similarity or part of changing of network during meta-sampling
psi	Integer number. Size of each Voronoi diagram or number of areas/points in the Voronoi diagrams
t	Integer number of trees in the Isolation Forest
param	or par.sim - data frame of parameters of the model
talkative	Logical parameter to print or do not print messages
check_pos_def	Logical parameter to check the Gram matrix is positive definite or do not check
iKernelABC	Result of function iKernelABC
n_best	Integer number of the best tracer bullets / points to consider them at the next algorithmic step
cores	Number of available cores for parallel computing

n_bullets	Integer number of tracer bullets / additional points between the TWO most distant points
halfwidth	Criterion to choose the best tracer points like: if similarity_of_point >= halfwidth then it is the point to be included to the poool of the best points
max_iteration	Maximal number of iteration in the function
save_web	Logical to save or do not save network during meta-sampling

Value

restrict_data() returns the list of:
par.sim - restricted parameters which are close to observation point
stat.sim - restricted stat.sim which are close to observation point

sampler_MaxWiK() returns the list:
results - results of simulations;
best - the best value of parameter;
MSE_min - minimum of MSE;
number_of_iterations - number of iterations;
time - time of sampling in seconds.

The function iKernelABC() returns the list of :

- kernel_mean_embedding is a maxima weighted kernel mean embedding (mapping) related to the observation point;
- parameters_Matrix_Voronoi is a matrix of information about Voronoi trees (rows - trees, columns - Voronoi points/areas IDs) for parameters data set;
- parameters_Matrix_iKernel is a matrix of of all points of PARAMETERS in a Hilbert space (rows - points, columns - isolation trees);
- Hilbert_weights is a weights in Hilbert space to get maxima weighted kernel mean embedding for parameters_Matrix_iKernel;
- Matrix_iKernel is a matrix of all points of simulations in a Hilbert space (rows - points, columns - isolation trees);
- iFeature_point is a feature embedding mapping for the OBSERVATION point;
- similarity is a vector of similarities between the simulation points and observation point;
- Matrix_Voronoi is a matrix of information about Voronoi trees (rows - trees, columns - Voronoi points/areas IDs);
- t is a number of trees in the Isolation Forest;
- psi is a number of areas/points in the Voronoi diagrams

Get_iKernel_estimation() returns list of:

- iKernel_ABC - parameter estimation based on isolation kernel / weighted sum;
- K2_ABC_iKernel - parameter estimation based on K2-ABC method with matrix of isolation kernel.

adjust_psi_t() returns adjusted hyper parameters psi and t as a data.frame with set of pair psi_t

The function spiderweb() returns the list of the next objects:

- input.parameters the list of all the input parameters for Isolation Kernel ABC method;
- par.best that is data frame of one point that is the best from all the generated tracer points;
- par.top that is data frame of n_best points that are the top from all the generated tracer points;
- sim.top that is numeric vecor of similarities of the top points;
- sim.best that is numeric value of the similarity of the best tracer point; tracers_all that is data frame of all the generated tracer points;
- sim.tracers_all that is numeric vector of similarities of all the generated tacer points;
- iKernelABC that is result of the function `iKernelABC()` given on `input parameters`.

The function `spiderweb()` returns the list of the next objects:

- input.parameters that is the list of all the input parameters for Isolation Kernel ABC method;
- iteration that is iteration value when algorithm stopped;
- network that is network points when algorithm stopped;
- par.best that is data frame of one point that is the best from all the generated tracer points;
- sim.best that is numeric value of the similarity of the best tracer point;
- iKernelABC that is result of the function `iKernelABC()` given on `input parameters`;
- spiderweb that is the list of all the networks during the meta-sampling.

The function `spiderweb()` returns the list of the next objects:

- input.parameters that is the list of all the input parameters for Isolation Kernel ABC method;
- iteration that is iteration value when algorithm stopped;
- network that is network points when algorithm stopped;
- par.best that is data frame of one point that is the best from all the generated tracer points;
- sim.best that is numeric value of the similarity of the best tracer point;
- iKernelABC that is result of the function `iKernelABC()` given on `input parameters`;
- spiderweb that is the list of all the networks during the meta-sampling.

Functions

- `restrict_data()`: Function to restrict data in the size to accelerate the calculations
- `sampler_MaxWiK()`: Function to generate parameters and simulate a model based on MaxWiK algorithm
- `Get_iKernel_estimation()`: function to get parameter estimation based on isolation kernel
- `adjust_psi_t()`: Function to adjust hyper parameters `psi` and `t` for isolation kernel ABC
- `spiderweb_old()`: The function to get the best value of parameter corresponding to Maxima Weighted Isolation Kernel mapping which is related to an observation point
- `spiderweb()`: The function to get the best value of parameter corresponding to Maxima Weighted Isolation Kernel mapping which is related to an observation point
- `spiderweb_slow()`: The function to get the best value of parameter corresponding to Maxima Weighted Isolation Kernel mapping which is related to an observation point

Examples

```
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
```

`Results_toy_experiments` *Data frame with results of function* `experiment_models()`

Description

A dataset containing results of function `experiment_models()` with next input:
`Results_toy_experiments = experiment_models( file_name = '../Results_ALL.txt',`
`models = c( 'Gaussian', 'Linear' ),`
`dimensions = (1:10)*2,`
`stochastic_terms = c( 0, 1, 5, 10, 20, 30 ),`
`rng = c( 0,10 ),`
`restrict_points_number = 300 ) )`
Please, pay attention it will take 4-6 hours for 4 cores.

To analyze these data, please, use:
`RES_methods = analyze_experiments( Results_toy_experiments, file_to_save = '../ggplot.pdf' )`
that gives the best methods for each case and statistical data of simulations

Usage

```
Results_toy_experiments
```

Format

A data frame with 12960 rows and 8 variables:

method_name Name of a method for calculation

kernel_name Name of using kerel or ” if not applicapable

model_name Name of a model, can be ’Gaussian’ or ’Linear’

dimension Dimension of the toy task

stochastic_term Stochastic term in a function of a model

MSE Mean square error

running_time Running time in seconds

iteration Iteration number

`sampler_method` *Function to call a method to get MSE*

Description

Function to call a method to get MSE

Usage

```
sampler_method(
  stat.obs,
  stat.sim,
  par.sim,
  model,
  method_name,
  kernel_name,
  arg0 = list(),
  size = 500,
  nmax = 30,
  G,
  model_name,
  dimension,
  stochastic_term,
  par.truth
)

experiment_samplers(
  file_name = "./output.txt",
  model_name = "Gaussian",
  dimension = 6,
  stochastic_term = 5,
  rng = c(0, 10),
  restrict_points_number = 500,
  nmax = 30
)

sampler_all_methods(
  model_name,
  dimension,
  stochastic_term,
  stat.obs,
  stat.sim,
  par.sim,
  G,
  par.truth,
  cores = 4,
  nmax
)

Get_call(
  method_name,
```

```
  kernel_name = "",
  model_name,
  dimension,
  stochastic_term,
  iteration,
  stat.obs,
  stat.sim,
  par.sim,
  G = NULL,
  par.truth
)

Get_call_all_methods(
  model_name,
  dimension,
  stochastic_term,
  iterations,
  stat.obs,
  stat.sim,
  par.sim,
  G,
  par.truth,
  cores = 4
)
```

Arguments

`stat.obs`	Data frame of statistics of observation point
`stat.sim`	Data frame of statistics of simulations
`par.sim`	Data frame of parameters
`model`	is a function to get output of simulation during sampling
`method_name`	Name of a method
`kernel_name`	Name of kernel function
`arg0`	is a list with arguments for a model function, so that arg0 is NOT changed during sampling
`size`	Number of point to restrict original dataset
`nmax`	is maximal number of iterations
`G`	Matrix of similarities for K2-ABC based on isolation kernel
`model_name`	Name of a model
`dimension`	Dimension of the model
`stochastic_term`	A number (usually in the range `[0,1]`) of stochastic term in the dataset `stat.sim`
`par.truth`	Truth parameter value to check result of estimation
`file_name`	File to save results
`rng`	Range of points for each dimension, by default `rng = range(0,10)`
`restrict_points_number`	Number of points which will be used in each method and that are close to observation point
`cores`	Number of cores for parallel calculation with iterations
`iteration`	Iteration number of trial with the same model and other parameters

Value

`sampler_method()` returns the list:
results - results of simulations;
best - the best value of parameter;
MSE_min - minimum of MSE;
number_of_iterations - number of iterations;
time - time of sampling in seconds.

`experiment_samplers()` returns data frame with results of experiment for sampling using all the methods

`sampler_all_methods()` returns data.frame with MSE for all defined methods

`Get_call()` returns the list:
method_name = method_name,

- kernel_name,
- model_name,
- stochastic_term,
- MSE,
- running_time,
- iteration.

`Get_call_all_methods()` returns data.frame with MSE for all defined methods

Functions

- `sampler_method()`: Function to generate parameters and simulate a model based on MaxWiK algorithm
- `experiment_samplers()`: Make experiments with sampling for all the methods and one case of toy model and dimension
- `sampler_all_methods()`: Function to call all the methods to get estimation of parameter and MSE
- `Get_call_all_methods()`: Function to call all the methods to get estimation of parameter and MSE

Examples

```
NULL
NULL
NULL
NULL
NULL
```

simulation_example *Example of simulation for lazy start*

Description

Example of simulation for lazy start

Usage

```
simulation_example(
  verbose = TRUE,
  to_plot = TRUE,
  seed = NA,
  model = c("Gaussian", "linear")[2],
  d = 2,
  x0 = c(3, 4),
  probability = TRUE,
  n = 1000,
  r = range(0, 10),
  psi = 8,
  t = 8,
  restrict_points_number = 300
)

simulation_example_many_psi_t(
  verbose = TRUE,
  to_plot = TRUE,
  seed = NA,
  model = c("Gaussian", "linear")[2],
  d = 2,
  x0 = c(3, 4),
  probability = TRUE,
  n = 1000,
  r = range(0, 10),
  psi_t = data.frame(psi = c(4, 4, 8, 8, 8, 8, 10, 10), t = c(8, 20, 6, 8, 16, 20, 6,
    12)),
  restrict_points_number = 300,
  cores = 4
)

get_Spider_MAP(
  stat.sim,
  par.sim,
  stat.obs,
  restrict_points_number = 300,
  cores = 4,
  n_best = 8
)

plot_3d_similarities(sim, r = range(0, 10), n = 12)
```

```
plot_3d_net_similarities(simnet, r = range(0, 10), n = 20)
```

Arguments

verbose	Logical type to show or do not show messages during execution
to_plot	Logical type to plot or do not plot graphical results of a simulation
seed	Numeric type to set seed for a simulation, if seed = NA (by default) then it will be skipped
model	Name of the toy model, can be 'Gaussian' or 'linear' only
d	Integer number of dimension of the model
x0	Truth value of parameter corresponding to observation point
probability	Logical to apply the probabilistic distribution to input data generation
n	Integer number of points for each axes to get net for plotly 3D graph
r	Range of parameters to plot, by default r = range(0, 10)
psi	Integer number. Size of each Voronoi diagram or number of areas/points in the Voronoi diagrams
t	Integer number of trees in the Isolation Forest
restrict_points_number	Maximal number of points in the data sets to get MAP
psi_t	Data.frame with different set of psi and t hyperparameters
cores	Number of cores for parallel calculation
stat.sim	Summary statistics of the simulations (model output)
par.sim	• data frame of parameters of the model
stat.obs	Summary statistics of the observation point
n_best	Number of best psi_t pairs adjusted for MaxWiK algorithm
sim	Results of the function simulation_example
simnet	Results of the function simulation_example_many_psi_t

Value

List of results of simulation with default values for all the parameters

List of results of simulation with default values for all the parameters

Maximum A Posteriori of meta-sampling distribution of parameters

Plot of 3D to show the similarity of each point of the parameter space

Many 3D plots to show the similarity of each point of the parameter space for each set of hyperparameters psi and t

Functions

- simulation_example_many_psi_t(): Example of simulation for lazy start and different psi / t hyperparameters
- get_Spider_MAP(): Function to get MAP of SpiderWeb algorithm based on different psi / t hyperparameters
- plot_3d_similarities(): Function to plot whole space of parameters related to similarity of the observation point:
- plot_3d_net_similarities(): Function to plot whole space of parameters related to similarity of the observation point for each set of hyperparameters psi and t

Examples

```
NULL
# it takes a time for a simulation and then it will demonstrates results, \cr
# so, please, wait for a while
sim = simulation_example( verbose = FALSE , to_plot = FALSE )
NULL
# it takes a time for a simulation and then it will demonstrates results, \cr
# so, please, wait for a while
sim = simulation_example_many_psi_t( verbose = FALSE , to_plot = FALSE )
NULL
# it takes a time for a simulation and then it will demonstrates results, \cr
# so, please, wait for a while
sim = simulation_example_many_psi_t( verbose = FALSE , to_plot = FALSE )
NULL
NULL
```

---

sudoku *The function to get the best tracer bullets related to kernel mean embedding*

Description

The function `sudoku()` allows to get the best tracer bullets related to kernel mean embedding. The calculation performs ONLY for parameters dataset DT = par.sim. This function performs a heuristic algorithm to seek a space/area related to the feature mapping in Hilbert space for the dataset of the parameters.
The main idea of the algorithm is just:

1. Generate points between the centers of Voronoi diagrams related to the Maxima weighted feature mapping based on Isolation Kernel
2. Following strategy to puzzle out of SUDOKU: delete all points that do not match feature mapping
3. Output: The remaining points should be corresponding to the feature mapping.

The function `get_pairs_of_data_frame()` is used to get pairs of points from the Data Frame that is the most distant each other. In other words, the algorithm seeks the most distant coupled point to each point from the data frame

The function `generate_points_between_two_points()` is used to generate points between two given points

The function `get_tracer_bullets()` is used to to get 'tracer bullets' or tracer points generated between all the pairs of the most distant points

Usage

```
sudoku(DT, iKernelABC, n_bullets = 20, n_best = 10, halfwidth = 0.5)

get_pairs_of_data_frame(DF)
```

```
generate_points_between_two_points(pair, n = 10)

get_tracer_bullets(DF, n_bullets = 20)
```

Arguments

DT	Whole dataset of parameters
iKernelABC	Result of calculations based on Isolation Kernel ABC that can be gotten by the function iKernelABC()
n_bullets	Integer number of tracer points between each pair of points from DF
n_best	Integer number of the best tracer bullets / points to consider them at the next algorithmic step
halfwidth	Criterion to choose the best tracer points like: if similarity_of_point >= halfwidth then it is the point to be included to the pool of the best points
DF	Data frame of oints that is used for generation of tracer points, so it is usually a subset of points corresponding to Voronoi sites/seeds
pair	Data frame of two points
n	Integer number of points that should be located between two input points

Value

The function sudoku() returns the list of next objects:

- tracer_bullets that is all the points generated during the run of the algorithm,
- criterion that is a value of the similarity that is used to choose the best tracer points,
- best_tracer_bullets that is the best tracer points that have similarity more or equal than **criterion** value,
- surroundings_best_points that is the best tracer points that have similarity more or equal than **halfwidth** value,
- feature_tracers that is results of the function get_voronoi_feature_PART_dataset() applied to the new tracer points,
- similarity_to_mean that is numeric vector of similarities of all the tracers points.

The function get_pairs_of_data_frame() returns the list of the pairs of points

The function generate_points_between_two_points() returns data frame of generated points between two given points, including given points as the first and the last rows

The function get_tracer_bullets() returns data frame of generated tracer points

Functions

- get_pairs_of_data_frame(): The function to get pairs from Data Frame
- generate_points_between_two_points(): The function to generate points between the pair of given points
- get_tracer_bullets(): The function to get 'tracer bullets' or tracer points

Examples

```
NULL
```

```
NULL
NULL
NULL
```

Index

∗ **datasets**
Results_toy_experiments, 32

add_new_point_iKernel *(*get_voronoi_feature*)*, 14
adjust_ABC_tolerance *(*K2_ABC*)*, 17
adjust_Gram, 2
adjust_K2_ABC *(*K2_ABC*)*, 17
adjust_K2_ABC_iKernel *(*K2_ABC*)*, 17
adjust_psi_t *(*restrict_data*)*, 26
analyze_experiments, 3

check_numeric_format, 4
check_packages, 4
check_pkg, 5
check_positive_definite *(*get_inverse_GRAM*)*, 9
copy_pipelines, 5

experiment_models, 6
experiment_samplers *(*sampler_method*)*, 33

Gaussian_model, 7
gen_colors, 8
generate_points_between_two_points *(*sudoku*)*, 38
Get_call *(*sampler_method*)*, 33
Get_call_all_methods *(*sampler_method*)*, 33
Get_iKernel_estimation *(*restrict_data*)*, 26
get_inverse_GRAM, 9
get_kernel_mean_embedding, 10
Get_MAP, 10
get_network_from_simnet *(*get_par_best_from_simnet*)*, 12
get_pairs_of_data_frame *(*sudoku*)*, 38
get_par_best_from_simnet, 12
Get_parameter, 11
get_Spider_MAP *(*simulation_example*)*, 36
get_spiderweb_from_simnet *(*get_par_best_from_simnet*)*, 12
GET_SUBSET, 13
get_subset_of_feature_map, 13
get_tracer_bullets *(*sudoku*)*, 38
get_voronoi_feature, 14
get_voronoi_feature_PART_dataset, 15
get_weights_iKernel *(*iKernel*)*, 15
GRAM_iKernel *(*iKernel*)*, 15

iKernel, 15
iKernel_point_dataset *(*iKernel*)*, 15
iKernelABC *(*restrict_data*)*, 26

K2_ABC, 17

linear_model *(*Gaussian_model*)*, 7

Mean_iKernel_parameters, 18
model, 19
MSE_parameters *(*MSE_sim*)*, 20
MSE_sim, 20

norm_vec, 21
norm_vec_sq *(*norm_vec*)*, 21

plot_2D, 21
plot_2D_lines, 22
plot_3d_net_similarities *(*simulation_example*)*, 36
plot_3d_similarities *(*simulation_example*)*, 36
plot_sudoku_2D, 24
plot_web_2D *(*plot_sudoku_2D*)*, 24

read_file, 26
restrict_data, 26
Results_toy_experiments, 32

sampler_all_methods *(*sampler_method*)*, 33
sampler_MaxWiK *(*restrict_data*)*, 26
sampler_method, 33
simulation_example, 36
simulation_example_many_psi_t *(*simulation_example*)*, 36
spiderweb *(*restrict_data*)*, 26
spiderweb_old *(*restrict_data*)*, 26
spiderweb_slow *(*restrict_data*)*, 26
sudoku, 38