Package ‘ExaGeoStatCPP’

April 21, 2024

Type Package

Title R Package Demonstrates the R/C++ Language Interface for Exascale GeoStatistics software

Version 1.1.0

Date 2024-01-14

Author Mahmoud ElKarargy [aut, cph], Sameh Abdulah [cre, cph], KAUST King Abdullah University of Science and Technology [fnd, cph], Brightskies [cph]

Maintainer Sameh Abdulah <sameh.abdulah@kaust.edu.sa>

Description An R-Interface for ExaGeoStatCPP: a parallel high performance unified framework for geostatistics on manycore systems. Its abbreviation stands for Exascale Geostatistics. The framework aims at optimizing the likelihood function for a given spatial data to provide an efficient way to predict missing observations. The framework targets many-core systems: clusters of CPUs and GPUs.

License GPL (>= 3)

Imports assertthat (>= 0.2.1), MASS, methods, Rcpp (>= 1.0.9)

Depends R (>= 3.5.0), assertthat (>= 0.2.1), MASS

RoxygenNote 7.2.3

SystemRequirements C++ (>= 11), lapacke (https://github.com/xianyi/OpenBLAS/releases)

NeedsCompilation yes

OS_type unix

URL https://www.github.com/ecrc/ExaGeoStatCPP

BugReports https://github.com/ecrc/ExaGeoStatCPP/issues

Encoding UTF-8

R topics documented:

`ExaGeoStatData` *ExaGeoStatData Class*

Description

The ExaGeoStatData class represents a data component in the ExaGeoStat system, that manages geo-statistical data with functions for location and descriptor manipulation. It is initialized with the size and dimension of the data.

Value

An object of class `ExaGeoStatData` representing a data component with the specified size and dimension.

Constructor

`ExaGeoStatData` Creates a new instance of the `ExaGeoStatData` class. `ExaGeoStatData(size,dimension)`

`size` An integer representing the size of the locations data.

`dimension` A string representing the dimensions of the data. - available dimension ("2D", "3D", "ST")

Examples

```
problem_size <- 4
dimension = "3D"
empty_data <- new(Data, problem_size, dimension)
```

`ExaGeoStatHardware` *ExaGeoStatHardware Class*

Description

The ExaGeoStatHardware class represents a hardware component in the ExaGeoStat system. It is initialized with computation mode, and two integers representing number of CPU cores and number of GPU cores.

Value

An object of class `ExaGeoStatHardware` representing a hardware component with the specified component and number of CPU cores and GPU cores.

Constructor

`ExaGeoStatHardware` Creates a new instance of the `ExaGeoStatHardware` class. `ExaGeoStatHardware(computation,num_`

`computation` A string specifying the computation method, either "exact" or "dst" or "tlr".

`num_of_cpus` An integer representing number of CPU cores.

`num_of_gpus` An integer representing number of GPU cores.

Methods

finalize_hardware: `finalize_hardware()` Manually finalizes the hardware by resetting the context.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

hardware$finalize_hardware()
```

`fisher` *Compute the Fisher information matrix for a given data and theta vector*

Description

This function computes the Fisher information matrix for a given dataset and theta vector, using a specified kernel and distance metric. It also allows for the inclusion of missing values and the specification of data dimensions.

Usage

```
fisher(kernel, distance_matrix = "euclidean", estimated_theta, dts, lts = 0, dimension = "2D", train_da
```

Arguments

`kernel` A string specifying the kernel to use - available kernels ("BivariateMaternFlexible", "BivariateMaternParsimonious", "BivariateSpacetimeMaternStationary", "TrivariateMaternParsimonious", "UnivariateExpNonGaussian", "UnivariateMaternDbeta", "UnivariateMaternDdbetaBeta", "UnivariateMaternDdbetaNu", "UnivariateMaternDdnuNu", "UnivariateMaternDdsigmaSquare", "UnivariateMaternDdsigmaSquareBeta", "UnivariateMaternDdsigmaSquareNu", "UnivariateMaternDnu", "UnivariateMaternDsigmaSquare", "UnivariateMaternNonGaussian", "UnivariateMaternNuggetsStationary", "UnivariateMaternStationary", "UnivariatePowExpStationary", "UnivariateSpacetimeMaternStationary", "bivariate_matern_flexible", "bivariate_matern_parsimonious", "bivariate_spacetime_matern_stationary", "trivariate_matern_parsimonious", "univariate_exp_non_gaussian", "univariate_matern_dbeta", "univariate_matern_ddbeta_beta", "univariate_matern_ddbeta_nu", "univariate_matern_ddnu_nu", "univariate_matern_ddsigma_square", "univariate_matern_ddsigma_square_beta", "univariate_matern_ddsigma_square_nu", "univariate_matern_dnu", "univariate_matern_dsigma_square", "univariate_matern_non_gaussian", "univariate_matern_nuggets_stationary", "univariate_matern_stationary", "univariate_pow_exp_stationary", "univariate_spacetime_matern_stationary")

`distance_matrix` A string specifying the distance metric, either "euclidean" or "great_circle". Default is "euclidean".

`estimated_theta` A list of estimated theta parameters.

`dts` A numeric value representing the time step size.

`lts` A numeric value representing the length step size. Default is 0.

`dimension` A string specifying the data dimension, either "2D" or "3D". Default is "2D".

`train_data` A numeric vector contains the locations and z measurements for training

`test_data` A numeric vector contains the locations for testing.

Value

A vector containing the Fisher information matrix elements.

Examples

```
dimension = "2D"
ncores <- 1
ngpus <- 0
dts <- 2
kernel <- "univariate_matern_stationary"
estimated_theta <- c(1,0.1,0.5)
computation <- "exact"

hardware <- new(Hardware, computation, ncores, ngpus)
```

```
z_value <- c(-1.272336140360187606, -2.590699695867695773, 0.512142584178685967, -0.163880452049749520)
locations_x <- c(0.092042420080872822, 0.193041886015106440, 0.330556191348134576, 0.181612878614480805)
locations_y <- c(0.928648813611047563, 0.103883421072709245, 0.135790035858701447, 0.434683756771190977)

test_x <- c(0.347951, 0.62768)
test_y <- c(0.806332, 0.105196)

fisher_matrix <- fisher(train_data=list(locations_x, locations_y, z_value), test_data=list(test_x, test_y), kerne
```

`get_locationsX` *Get X Locations*

Description

Retrieves X coordinates of locations from ExaGeoStatData object.

Usage

```
get_locationsX(data)
```

Arguments

`data`	A list of ExaGeoStatData that contains the locations.

Value

A numeric vector of X locations.

Examples

```
ncores <- 1
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

dimension = "2D"
problem_size <- 4
empty_data <- new(Data, problem_size, dimension)

dts <- 2
kernel <- "univariate_matern_stationary"
initial_theta <- c(1,0.1,0.5)
exageostat_data <- simulate_data(kernel=kernel, initial_theta=initial_theta, problem_size=problem_size, dts=dts,

x <- get_locationsX(data=exageostat_data)
```

get_locationsY *Get Y Locations*

Description

This function retrieves the Y locations from the provided data.

Usage

```
get_locationsY(data)
```

Arguments

data   A list of ExaGeoStatData that contains the locations.

Value

A numeric vector of Y locations.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

dimension = "2D"
problem_size <- 4
empty_data <- new(Data, problem_size, dimension)

dts <- 2
kernel <- "univariate_matern_stationary"
initial_theta <- c(1,0.1,0.5)
exageostat_data <- simulate_data(kernel=kernel, initial_theta=initial_theta, problem_size=problem_size, dts=dts,

y <- get_locationsY(data=exageostat_data)
```

get_locationsZ *Get Z Locations*

Description

Retrieves Z coordinates of locations from ExaGeoStatData object.

Usage

```
get_locationsZ(data)
```

Arguments

`data` A list of ExaGeoStatData that contains the locations.

Value

A numeric vector of Z locations.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

dimension = "3D"
problem_size <- 4
empty_data <- new(Data, problem_size, dimension)

dts <- 2
kernel <- "univariate_matern_stationary"
initial_theta <- c(1,0.1,0.5)
exageostat_data <- simulate_data(kernel=kernel, initial_theta=initial_theta, problem_size=problem_size, dts=dts,

z <- get_locationsZ(data=exageostat_data)
```

---

get_Z_measurement_vector

Get descriptive Z values from ExaGeoStat data

Description

Retrieves descriptive Z values from ExaGeoStat data based on type.

Usage

```
get_Z_measurement_vector(data,type)
```

Arguments

`data` A list of ExaGeoStatData that contains the locations.

`type` A string specifying the type of descriptor value to retrieve (e.g., "Chameleon", "HiCMA").

Value

A numeric vector of descriptive Z values.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

dimension = "3D"
problem_size <- 4
empty_data <- new(Data, problem_size, dimension)

dts <- 2
kernel <- "univariate_matern_stationary"
initial_theta <- c(1,0.1,0.5)
exageostat_data <- simulate_data(kernel=kernel, initial_theta=initial_theta, problem_size=problem_size, dts=dts,

Z <- get_Z_measurement_vector(data=exageostat_data, type="chameleon")
```

---

idw — *This function performs IDW interpolation for a given dataset and theta vector.*

Description

This function performs Inverse Distance Weighting (IDW) interpolation for a given dataset and theta vector.

Usage

```
idw(kernel, distance_matrix = "euclidean", estimated_theta, dts, lts = 0, dimension = "2D", train_data,
```

Arguments

`kernel` A string specifying the kernel to use - available kernels ("BivariateMaternFlexible", "BivariateMaternParsimonious", "BivariateSpacetimeMaternStationary", "TrivariateMaternParsimonious", "UnivariateExpNonGaussian", "UnivariateMaternDbeta", "UnivariateMaternDdbetaBeta", "UnivariateMaternDdbetaNu", "UnivariateMaternDdnuNu", "UnivariateMaternDdsigmaSquare", "UnivariateMaternDdsigmaSquareBeta", "UnivariateMaternDdsigmaSquareNu", "UnivariateMaternDnu", "UnivariateMaternDsigmaSquare", "UnivariateMaternNonGaussian", "UnivariateMaternNuggetsStationary", "UnivariateMaternStationary", "UnivariatePowExpStationary", "UnivariateSpacetimeMaternStationary", "bivariate_matern_flexible", "bivariate_matern_parsimonious", "bivariate_spacetime_matern_stationary", "trivariate_matern_parsimonious", "univariate_exp_non_gaussian", "univariate_matern_dbeta", "univariate_matern_ddbeta_beta", "univariate_matern_ddbeta_nu", "univariate_matern_ddnu_nu", "univariate_matern_ddsigma_square", "univariate_matern_ddsigma_square_beta", "univariate_matern_ddsigma_square_nu", "univariate_matern_dnu", "univariate_matern_dsigma_square" "univariate_matern_non_gaussian", "univariate_matern_nuggets_stationary", "univariate_matern_stationary", "univariate_pow_exp_stationary", "univariate_spacetime_matern_stationary')

distance_matrix
: A string specifying the distance metric, either "euclidean" or "great_circle". Default is "euclidean"

estimated_theta
: A list of estimated theta parameters

dts
: A numeric value representing the time step size

lts
: A numeric value representing the length step size. Default is 0

dimension
: A string specifying the data dimension, either "2D" or "3D". Default is "2D"

train_data
: A numeric vector contains the locations and z measurements for training

test_data
: A numeric vector contains the locations for testing.

test_measurements
: A numeric vector contains the z measurements for testing.

Value

A vector containing the IDW error.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

problem_size <- 4
dimension = "2D"
dts <- 2
kernel <- "univariate_matern_stationary"
estimated_theta <- c(1,0.1,0.5)

z_value <- c( -1.272336140360187606, -2.590699695867695773, 0.512142584178685967, -0.163880452049749520)
locations_x <- c(0.193041886015106440, 0.330556191348134576, 0.181612878614480805, 0.370473792629892440)
locations_y <- c(0.103883421072709245, 0.135790035858701447, 0.434683756771190977, 0.400778210116731537)
test_x <- c(0.347951, 0.62768)
test_y <- c(0.806332, 0.105196)
test_measurements = c(-1.05428, -1.47441)

idw_error = idw(kernel=kernel, estimated_theta=estimated_theta, dts=dts, train_data=list(locations_x, locations_
```

`mloe_mmom` *Mean Misspecification of the Mean Square Error (MMOM) and Mean Loss of Efficiency (MLOE) using exact method.*

Description

This function calculates Mean Misspecification of the Mean Square Error (MMOM) and Mean Loss of Efficiency (MLOE).

Usage

```
mloe_mmom(kernel, distance_matrix="euclidean", estimated_theta, true_theta, dts, lts=0, dimension="2D
```

Arguments

kernel A string specifying the kernel to use - available kernels ("BivariateMaternFlexible", "BivariateMaternParsimonious", "BivariateSpacetimeMaternStationary", "TrivariateMaternParsimonious", "UnivariateExpNonGaussian", "UnivariateMaternDbeta", "UnivariateMaternDdbetaBeta", "UnivariateMaternDdbetaNu", "UnivariateMaternDdnuNu", "UnivariateMaternDdsigmaSquare", "UnivariateMaternDdsigmaSquareBeta", "UnivariateMaternDdsigmaSquareNu", "UnivariateMaternDnu", "UnivariateMaternDsigmaSquare", "UnivariateMaternNonGaussian", "UnivariateMaternNuggetsStationary", "UnivariateMaternStationary", "UnivariatePowExpStationary", "UnivariateSpacetimeMaternStationary", "bivariate_matern_flexible", "bivariate_matern_parsimonious", "bivariate_spacetime_matern_stationary", "trivariate_matern_parsimonious", "univariate_exp_non_gaussian", "univariate_matern_dbeta", "univariate_matern_ddbeta_beta", "univariate_matern_ddbeta_nu", "univariate_matern_ddnu_nu", "univariate_matern_ddsigma_square", "univariate_matern_ddsigma_square_beta", "univariate_matern_ddsigma_square_nu", "univariate_matern_dnu", "univariate_matern_dsigma_square" "univariate_matern_non_gaussian", "univariate_matern_nuggets_stationary", "univariate_matern_stationary", "univariate_pow_exp_stationary", "univariate_spacetime_matern_stationary")

distance_matrix A string specifying the distance metric, either "euclidean" or "great_circle". Default is "euclidean"

estimated_theta A list of estimated theta parameters

true_theta A list of truth theta parameters

dts A numeric value representing the time step size

lts A numeric value representing the length step size. Default is 0

dimension A string specifying the data dimension, either "2D" or "3D". Default is "2D"

train_data A numeric vector contains the locations and z measurements for training

test_data A numeric vector contains the locations for testing.

Value

A vector of MLOE/MMOM values

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

problem_size <- 4
dimension = "2D"
```

```
dts <- 2
kernel <- "univariate_matern_stationary"
estimated_theta <- c(1,0.1,0.5)
true_theta <- c(1.1,0.2,0.5)

z_value <- c(-1.272336140360187606, -2.590699695867695773, 0.512142584178685967, -0.163880452049749520)
locations_x <- c(0.092042420080872822, 0.193041886015106440, 0.330556191348134576, 0.181612878614480805)
locations_y <- c(0.928648813611047563, 0.103883421072709245, 0.135790035858701447, 0.434683756771190977)

test_x <- c(0.347951, 0.62768)
test_y <- c(0.806332, 0.105196)

result_mloe_mmom = mloe_mmom(train_data=list(locations_x, locations_y, z_value), test_data=list(test_x, test_y),
```

`model_data` *This function models data based on the provided computation method, kernel, distance matrix, and other parameters.*

Description

This function models data based on the provided computation method, kernel, distance matrix, and other parameters.

Usage

```
model_data(computation = "exact", kernel, distance_matrix = "euclidean", lb, ub, tol = 4, mle_itr, dts,
```

Arguments

computation	A string specifying the computation method, either "exact" or "dst" or "tlr". Default is "exact".
kernel	A string specifying the kernel to use - available kernels ("BivariateMaternFlexible", "BivariateMaternParsimonious", "BivariateSpacetimeMaternStationary", "TrivariateMaternParsimonious", "UnivariateExpNonGaussian", "UnivariateMaternDbeta", "UnivariateMaternDdbetaBeta", "UnivariateMaternDdbetaNu", "UnivariateMaternDdnuNu", "UnivariateMaternDdsigmaSquare", "UnivariateMaternDdsigmaSquareBeta", "UnivariateMaternDdsigmaSquareNu", "UnivariateMaternDnu", "UnivariateMaternDsigmaSquare", "UnivariateMaternNonGaussian", "UnivariateMaternNuggetsStationary", "UnivariateMaternStationary", "UnivariatePowExpStationary", "UnivariateSpacetimeMaternStationary", "bivariate_matern_flexible", "bivariate_matern_parsimonious", "bivariate_spacetime_matern_stationary", "trivariate_matern_parsimonious", "univariate_exp_non_gaussian", "univariate_matern_dbeta", "univariate_matern_ddbeta_beta", "univariate_matern_ddbeta_nu", "univariate_matern_ddnu_nu", "univariate_matern_ddsigma_square", "univariate_matern_ddsigma_square_beta", "univariate_matern_ddsigma_square_nu", "univariate_matern_dnu", "univariate_matern_dsigma_square", "univariate_matern_non_gaussian", "univariate_matern_nuggets_stationary", "univariate_matern_stationary", "univariate_pow_exp_stationary", "univariate_spacetime_matern_stationary")

distance_matrix
: A string specifying the distance metric, either "euclidean" or "great_circle". Default is "euclidean".

lb
: A numeric value representing the lower bound for the computation.

ub
: A numeric value representing the upper bound for the computation.

tol
: A numeric value specifying the tolerance for the computation. Default is 4.

mle_itr
: A numeric value specifying the maximum number of iterations for the computation.

dts
: A numeric value representing the time step size.

lts
: A numeric value representing the length step size. Default is 0.

dimension
: A string specifying the data dimension, either "2D" or "3D". Default is "2D".

band
: A numeric value Bandwidth for band matrices, applicable in certain computational kernels.. Default is 0.

max_rank
: A numeric value specifying the Maximum rank for low-rank approximations.. Default is 500.

data
: A list of data vectors. Default is ‘R_NilValue‘.

matrix
: A matrix object. Default is ‘R_NilValue‘.

x
: A numeric vector. Default is ‘R_NilValue‘.

y
: A numeric vector. Default is ‘R_NilValue‘.

z
: A numeric vector. Default is ‘R_NilValue‘.

Value

A vector containing the starting theta.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

dimension = "2D"
problem_size <- 4
empty_data <- new(Data, problem_size, dimension)

dts <- 2
kernel <- "univariate_matern_stationary"
lower_bound <- c(0.1,0.1,0.1)
upper_bound <- c(5,5,5)

z_value <- c( -1.272336140360187606, -2.590699695867695773, 0.512142584178685967, -0.163880452049749520)
locations_x <- c(0.193041886015106440, 0.330556191348134576, 0.181612878614480805, 0.370473792629892440)
locations_y <- c(0.103883421072709245, 0.135790035858701447, 0.434683756771190977, 0.400778210116731537)

theta <- model_data(kernel=kernel, lb=lower_bound, ub=upper_bound, mle_itr=10, dts=dts, matrix=z_value, x=locatio
```

predict_data *This function predicts data based on the provided kernel, distance matrix, estimated theta, and other parameters.*

Description

This function predicts data based on the provided kernel, distance matrix, estimated theta, and other parameters.

Usage

```
predict_data(kernel, distance_matrix = "euclidean", estimated_theta, dts, lts = 0, dimension = "2D", tr
```

Arguments

kernel	A string specifying the kernel to use - available kernels ("BivariateMaternFlexible", "BivariateMaternParsimonious", "BivariateSpacetimeMaternStationary", "TrivariateMaternParsimonious", "UnivariateExpNonGaussian", "UnivariateMaternDbeta", "UnivariateMaternDdbetaBeta", "UnivariateMaternDdbetaNu", "UnivariateMaternDdnuNu", "UnivariateMaternDdsigmaSquare", "UnivariateMaternDdsigmaSquareBeta", "UnivariateMaternDdsigmaSquareNu", "UnivariateMaternDnu", "UnivariateMaternDsigmaSquare", "UnivariateMaternNonGaussian", "UnivariateMaternNuggetsStationary", "UnivariateMaternStationary", "UnivariatePowExpStationary", "UnivariateSpacetimeMaternStationary", "bivariate_matern_flexible", "bivariate_matern_parsimonious", "bivariate_spacetime_matern_stationary", "trivariate_matern_parsimonious", "univariate_exp_non_gaussian", "univariate_matern_dbeta", "univariate_matern_ddbeta_beta", "univariate_matern_ddbeta_nu", "univariate_matern_ddnu_nu", "univariate_matern_ddsigma_square", "univariate_matern_ddsigma_square_beta", "univariate_matern_ddsigma_square_nu", "univariate_matern_dnu", "univariate_matern_dsigma_square', "univariate_matern_non_gaussian", "univariate_matern_nuggets_stationary", "univariate_matern_stationary", "univariate_pow_exp_stationary", "univariate_spacetime_matern_stationary')
distance_matrix	A string specifying the distance metric, either "euclidean" or "great_circle". Default is "euclidean"
estimated_theta	A list of estimated theta parameters
dts	A numeric value representing the time step size
lts	A numeric value representing the length step size. Default is 0
dimension	A string specifying the data dimension, either "2D" or "3D". Default is "2D"
train_data	A numeric vector contains the locations and z measurements for training
test_data	A numeric vector contains the locations for testing.

Value

A vector of predicted z values

Examples

```
ncores <- 2
ngpus <- 0
problem_size <- 4
dts <- 2
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)
kernel <- "univariate_matern_stationary"
estimated_theta <- c(1,0.1,0.5)

z_value <- c( -1.272336140360187606, -2.590699695867695773, 0.512142584178685967, -0.163880452049749520)
locations_x <- c(0.193041886015106440, 0.330556191348134576, 0.181612878614480805, 0.370473792629892440)
locations_y <- c(0.103883421072709245, 0.135790035858701447, 0.434683756771190977, 0.400778210116731537)
test_x <- c(0.347951, 0.62768)
test_y <- c(0.806332, 0.105196)

predict_data(train_data=list(locations_x, locations_y, z_value), test_data=list(test_x, test_y), kernel=kernel, 
```

---

simulate_data — *This function simulates data based on the provided computation method, kernel, distance matrix, and other parameters.*

Description

This function loads data into an ExaGeoStatData object using the provided configuration and computational settings.

Usage

```
simulate_data(kernel, initial_theta, distance_matrix = "euclidean", problem_size, seed = 0, dts, lts = 
```

Arguments

`kernel` A string specifying the kernel to use - available kernels ("BivariateMaternFlexible", "BivariateMaternParsimonious", "BivariateSpacetimeMaternStationary", "TrivariateMaternParsimonious", "UnivariateExpNonGaussian", "UnivariateMaternDbeta", "UnivariateMaternDdbetaBeta", "UnivariateMaternDdbetaNu", "UnivariateMaternDdnuNu", "UnivariateMaternDdsigmaSquare", "UnivariateMaternDdsigmaSquareBeta", "UnivariateMaternDdsigmaSquareNu", "UnivariateMaternDnu", "UnivariateMaternDsigmaSquare", "UnivariateMaternNonGaussian", "UnivariateMaternNuggetsStationary", "UnivariateMaternStationary", "UnivariatePowExpStationary", "UnivariateSpacetimeMaternStationary", "bivariate_matern_flexible", "bivariate_matern_parsimonious", "bivariate_spacetime_matern_stationary", "trivariate_matern_parsimonious", "univariate_exp_non_gaussian", "univariate_matern_dbeta", "univariate_matern_ddbeta_beta", "univariate_matern_ddbeta_nu", "univariate_matern_ddnu_nu", "univariate_matern_ddsigma_square", "univariate_matern_ddsigma_square_beta", "univariate_matern_ddsigma_square_nu", "univariate_matern_dnu", "univariate_matern_dsigma_square" "univariate_matern_non_gaussian", "univariate_matern_nuggets_stationary", "univariate_matern_stationary", "univariate_pow_exp_stationary", "univariate_spacetime_matern_stationary")

`initial_theta` A list of initial theta parameters.

`distance_matrix` A string specifying the distance metric, either "euclidean" or "great_circle". Default is "euclidean".

`problem_size` A numeric value representing the size of the problem to simulate.

`seed` A numeric value specifying the seed for random number generation. Default is 0.

`dts` A numeric value representing the time step size.

`lts` A numeric value representing the length step size. Default is 0.

`dimension` A string specifying the data dimension, either "2D" or "3D". Default is "2D".

`log_path` A string specifying the path for logging. Default is "".

`data_path` A string specifying the path for data storage. Default is "".

`observations_file` A string specifying the file name for observations. Default is "".

`recovery_file` A string specifying the file name for recovery. Default is "".

Value

A pointer to ExaGeoStatData object that contains the loaded data.

Examples

```
ncores <- 2
ngpus <- 0
computation <- "exact"
hardware <- new(Hardware, computation, ncores, ngpus)

dimension = "2D"
problem_size <- 4
empty_data <- new(Data, problem_size, dimension)

dts <- 2
kernel <- "univariate_matern_stationary"
initial_theta <- c(1,0.1,0.5)

exageostat_data <- simulate_data(kernel=kernel, initial_theta=initial_theta, problem_size=problem_size, dts=dts,
```

Index