Package 'Manifoldgstat'

March 22, 2019

Type Package

Title Kriging prediction for manifold-valued data.

Version 1.0.0

Description Inference and prediction for manifold-valued data analysis. This package provides a C++ implementation of the functions to create a model, for spatial dependent manifold valued data, in order to perform kriging. In each location, specified by a vector of coordinates ([lat,long], [x,y] or [x,y,z]), the datum is supposed to be a symmetric and positive definite matrix (possibly a correlation matrix). The algorithm exploits a projection of these data on a tangent space, where the tangent point is either provided by the user or computed as intrinsic mean of the data in input.

Depends R (>= 3.2.0), Rcpp (>= 0.12.16), RcppEigen (>= 0.3.3.4.0), plyr(>= 1.8.4)

LinkingTo Rcpp, RcppEigen

NeedsCompilation yes

SystemRequirements C++11

License What license is it under?

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

bootstrapVar *Compute the bootstrap variance*

Description

Compute the bootstrap variance

Usage

```
bootstrapVar(res.boot, res.aggr, K, metric_manifold)
```

Arguments

res.boot	A list of length B. Each field contains the predictions for the corresponding iteration
res.aggr	A list as long as the number of the prediction. Each field a single prediction
K	number of neighbourhood used in the anaysis
metric_manifold	metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot"

Value

It returns a vector, as long as the number of predictions, containing the the variance at the predicted locations

distance_manifold *Distance on the manifold*

Description

Compute the manifold distance between symmetric positive definite matrices

Usage

```
distance_manifold(data1, data2, metric_manifold = "Frobenius")
```

Arguments

data1	list or array [p,p,B1] of B1 symmetric positive definite matrices of dimension p*p. Or a single p*p matrix
data2	list or array [p,p,B2] of B2 symmetric positive definite matrices of dimension p*p. Or a single p*p matrix.
metric_manifold	metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot", "Correlation"

Details

If B2=B1 then the result is a vector of length B1=B2 containing in position i the manifold distance beetween data1[,,i] and data2[,,i]. Instead if B2=1 and B1!=1 the result is a vector of length B1 containing in position i the manifold distance between data1[,,i] and data2[,,1]

Value

A double or a vector of distances

Examples

```
data_manifold_model <- Manifoldgstat::rCov
distances <-distance_manifold(data_manifold_model, diag(2), metric_manifold = "Frobenius")
print(distances)
```

---

full_RDD *Perform full_RDD*

Description

Perform full_RDD

Usage

```
full_RDD(data_coords, data_val, K, grid, nk_min = 1, B = 100,
  suppressMes = F, tol = 1e-12, max_it = 100, n_h = 15,
  tolerance_intrinsic = 10^(-6), X = NULL, X_new = NULL,
  ker.width.intrinsic = 0, ker.width.vario = 1.5,
  graph.distance.complete, data.grid.distance, N_samples, p,
  aggregation_mean, aggregation_kriging, method.analysis = "Local mean",
  metric_manifold, metric_ts, model_ts, vario_model, distance = NULL)
```

Arguments

data_coords	N*2 or N*3 matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees
data_val	array [p,p,N] of N symmetric positive definite matrices of dimension p*p
K	number of neighborhood (i.e., centers) to sample at each iteration
grid	prediction grid
nk_min	minimum number of observations within a neighborhood
B	number of *divide* iterations to perform
suppressMes	{TRUE, FALSE} controls the level of interaction and warnings given
tol	tolerance for the main loop of model_kriging
max_it	maximum number of iterations for the main loop of model_kriging
n_h	number of bins in the empirical variogram
tolerance_intrinsic	tolerance for the computation of the intrinsic mean. Not needed if Sigma is provided

X
: matrix (N rows and unrestricted number of columns) of additional covariates for the tangent space model, possibly NULL

X_new
: matrix (with the same number of rows of new_coords) of additional covariates for the new locations, possibly NULL

ker.width.intrinsic
: parameter controlling the width of the Gaussian kernel for the computation of the local mean (if 0, no kernel is used)

ker.width.vario
: parameter controlling the width of the Gaussian kernel for the computation of the empirical variogram (if 0, no kernel is used)

graph.distance.complete
: N*N distance matrix (the [i,j] element is the length of the shortest path between points i and j)

data.grid.distance
: N*dim(grid)[1] distance matrix between locations where the datum has been observed and locations where the datum has to be predicted

N_samples
: number of samples

p
: dimension of the manifold matrices

aggregation_mean
: "Weighted" ...

aggregation_kriging
: "Weighted" if the prediction must be aggregated using different weights, "Equal" to use equal weights

method.analysis
: "Local mean" to predict just with the mean, "Kriging" to predict via Kriging procedure

metric_manifold
: metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot"

metric_ts
: metric used on the tangent space. It must be chosen among "Frobenius", "FrobeniusScaled", "Correlation"

model_ts
: type of model fitted on the tangent space. It must be chosen among "Intercept", "Coord1", "Coord2", "Additive"

vario_model
: type of variogram fitted. It must be chosen among "Gaussian", "Spherical", "Exponential"

distance
: type of distance between coordinates. It must be either "Eucldist" or "Geodist"

Value

According to the analysis chosen:

- If method.analysis = "Local mean" it returns a list with the following fields
  - resBootstrap On its turn it is a list consisting of
    - fmean It is a list with length B. Each field contains, for each new location, its prediction (at iteration b) obtained as intrinsic mean of the data inside the tile it belongs to
    - kervalues_mean Weights used for aggregating fmean
  - resAggregatedPredictions, for each new location, obtained aggregating fmean using kervalues_mean as weights

- If `method.analysis` = "Kriging" it returns a list with the following fields
  - `resBootstrap` On its turn it is a list consisting of
    - `fmean` It is a list with length `B`. Each field contains, for each new location, its prediction (at iteration `b`) obtained as intrinsic mean of the data inside the tile it belongs to
    - `fpred` It is a list with length `B`. Each field contains, for each new location, the prediction (at iteration `b`) obtained using Kriging
    - `kervalues_mean` Weights used for aggregating `fmean`
    - `kervalues_krig` Weights used for aggregating `fpred`
    - `variofit` It is a list with length `B`. Each field contains, for each tile, the parameters of the fitted variogram
  - `resAggregated` Predictions, for each new location, obtained aggregating `fpred` using `kervalues_krig` as weights
  - `resLocalMean` Predictions, for each new location, obtained aggregating `fmean` using `kervalues_mean` as weights

---

`intrinsic_mean` *Intrinsic mean*

Description

Evaluate the intrinsic mean of a given set of symmetric positive definite matrices

Usage

```
intrinsic_mean(data, metric_manifold = "Frobenius",
  metric_ts = "Frobenius", tolerance = 1e-06,
  weight_intrinsic = NULL, weight_extrinsic = weight_intrinsic,
  tolerance_map_cor = 1e-06)
```

Arguments

`data`	list or array [`p,p,B`] of `B` symmetric positive definite matrices of dimension `p*p`
`metric_manifold`	metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot", "Correlation"
`metric_ts`	metric used on the tangent space. It must be chosen among "Frobenius", "FrobeniusScaled", "Correlation"
`tolerance`	tolerance for the computation of the intrinsic_mean
`weight_intrinsic`	vector of length `B` to weight the matrices in the computation of the intrinsic mean. If NULL a vector of ones is used
`weight_extrinsic`	vector of length `B` to weight the matrices in the computation of the extrinsic mean. If NULL weight_intrinsic are used
`tolerance_map_cor`	tolerance to use in the maps. Required only if `metric_manifold== "Correlation"`

Value

A matrix representing the intrinsic mean of the data

Examples

```
data_manifold_tot <- Manifoldgstat::fieldCov
Sigma <-intrinsic_mean(data_manifold_tot, metric_manifold = "Frobenius",
            metric_ts = "Frobenius")
print(Sigma)
```

`kriging` *Kriging prediction given the model*

Description

Given the GLS model kriging prediction on new location is performed.

Usage

```
kriging(GLS_model, coords, new_coords, model_ts = "Additive",
  vario_model = "Gaussian", metric_manifold = "Frobenius",
  X_new = NULL, distance = "Geodist", tolerance_map_cor = 1e-06)
```

Arguments

`GLS_model`	the object returned by `model_GLS`, or a list containing the fields: `Sigma` (tangent point), `beta` (vector of the beta matrices of the fitted model), `gamma_matrix` (`N*N` covariogram matrix), `residuals` (vector of the `N` residual matrices), `fitted_par_vario` (estimates of *nugget*, *sill-nugget* and *practical range*)
`coords`	`N*2` or `N*3` matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees
`new_coords`	matrix of coordinates for the new locations where to perform kriging
`model_ts`	type of model fitted on the tangent space. It must be chosen among "Intercept", "Coord1", "Coord2", "Additive"
`vario_model`	type of variogram fitted. It must be chosen among "Gaussian", "Spherical", "Exponential"
`metric_manifold`	metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot", "Correlation"
`X_new`	matrix (with the same number of rows of `new_coords`) of additional covariates for the new locations, possibly NULL
`distance`	type of distance between coordinates. It must be either "Eucldist" or "Geodist"
`tolerance_map_cor`	tolerance to use in the maps. Required only if `metric_manifold=="Correlation"`
`data_grid_dist_mat`	Matrix of dimension `N*M` of distances between data points and grid points. If not provided it is computed using `distance`

Details

The model provided is used to perform simple kriging on the tangent space in correspondence of the new locations. The estimates are then mapped to the manifold to produce the actual prediction.

Value

A list with a single field:

`prediction` vector of matrices predicted at the new locations

References

D. Pigoli, A. Menafoglio & P. Secchi (2016): Kriging prediction for manifold-valued random fields. Journal of Multivariate Analysis, 145, 117-131.

Examples

```
data_manifold_tot <- Manifoldgstat::fieldCov
data_manifold_model <- Manifoldgstat::rCov
coords_model <- Manifoldgstat::rGrid
coords_tot <- Manifoldgstat::gridCov
Sigma <- matrix(c(2,1,1,1), 2,2)

model = model_GLS(data_manifold = data_manifold_model, coords = coords_model, Sigma = Sigma,
              metric_manifold = "Frobenius", metric_ts = "Frobenius", model_ts = "Coord1",
                vario_model = "Spherical", n_h = 15, distance = "Eucldist", max_it = 100,
                  tolerance = 1e-7, plot = TRUE)
result = kriging (GLS_model = model, coords = coords_model, new_coords = coords_model,
              model_ts="Coord1", vario_model= "Spherical", metric_manifold = "Frobenius",
                  distance="Eucldist")
result_tot = kriging (GLS_model = model, coords = coords_model, new_coords = coords_tot,
               model_ts="Coord1", vario_model= "Spherical", metric_manifold = "Frobenius",
                      distance="Eucldist")

x.min=min(coords_tot[,1])
x.max=max(coords_tot[,1])
y.min=min(coords_tot[,2])
y.max=max(coords_tot[,2])
dimgrid=dim(coords_tot)[1]
radius = 0.02

par(cex=1.25)
plot(0,0, asp=1, col=fields::tim.colors(100), ylim=c(y.min,y.max), xlim=c(x.min, x.max),
     pch='', xlab='', ylab='', main = "Real Values")
for(i in 1:dimgrid){
  if(i %% 3 == 0)
     car::ellipse(c(coords_tot[i,1],coords_tot[i,2]), data_manifold_tot[,,i],
                    radius=radius, center.cex=.5, col='navyblue')
}
rect(x.min, y.min, x.max, y.max)

for(i in 1:250)
{ car::ellipse(c(coords_model[i,1],coords_model[i,2]), data_manifold_model[,,i],
               radius=radius, center.cex=.5, col='green')}
rect(x.min, y.min, x.max, y.max)
```

```
par(cex=1.25)
plot(0,0, asp=1, col=fields::tim.colors(100), ylim=c(y.min,y.max),xlim=c(x.min, x.max),
     pch='', xlab='', ylab='',main = "Predicted values")
for(i in 1:dimgrid){
  if(i %% 3 == 0)
     car::ellipse(c(coords_tot[i,1],coords_tot[i,2]), result_tot$prediction[[i]],
                  radius=radius, center.cex=.5, col='navyblue' )
}
rect(x.min, y.min, x.max, y.max)

for(i in 1:250)
{ car::ellipse(c(coords_model[i,1],coords_model[i,2]), result$prediction[[i]],
                radius=radius, center.cex=.5, col='red')}
rect(x.min, y.min, x.max, y.max)
```

`mixed_RDD` *Perform mixed_RDD*

Description

Perform mixed_RDD

Usage

```
mixed_RDD(data_coords, data_val, K, grid, nk_min = 1, B = 100,
  suppressMes = F, ker.width.intrinsic = 0, graph.distance.complete,
  data.grid.distance, N_samples, p, aggregation_mean, metric_ts,
  tol = 1e-12, max_it = 100, n_h = 15,
  tolerance_intrinsic = 10^(-6), X = NULL, X_new = NULL,
  create_pdf_vario = FALSE, pdf_parameters = NULL, metric_manifold,
  model_ts, vario_model, distance)
```

Arguments

`data_coords`	`N*2` or `N*3` matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees
`data_val`	array `[p,p,N]` of `N` symmetric positive definite matrices of dimension `p*p`
`K`	number of neighborhood (i.e., centers) to sample at each iteration
`grid`	prediction grid
`nk_min`	minimum number of observations within a neighborhood
`B`	number of \texitdivide iterations to perform
`suppressMes`	`{TRUE, FALSE}` controls the level of interaction and warnings given
`ker.width.intrinsic`	parameter controlling the width of the Gaussian kernel for the computation of the local mean (if 0, no kernel is used)
`graph.distance.complete`	`N*N` distance matrix (the [i,j] element is the length of the shortest path between points i and j)
`data.grid.distance`	`N*dim(grid)[1]` distance matrix between locations where the datum has been observed and locations where

N_samples	number of samples
p	dimension of the manifold matrices
aggregation_mean	"Weighted" ...
metric_ts	metric used on the tangent space. It must be chosen among "Frobenius", "FrobeniusScaled", "Correlation"
tol	tolerance for the main loop of model_kriging
max_it	maximum number of iterations for the main loop of model_kriging
n_h	number of bins in the empirical variogram
tolerance_intrinsic	tolerance for the computation of the intrinsic mean. Not needed if Sigma is provided
X	matrix (N rows and unrestricted number of columns) of additional covariates for the tangent space model, possibly NULL
X_new	matrix (with the same number of rows of `new_coords`) of additional covariates for the new locations, possibly NULL
create_pdf_vario	boolean. If `TRUE` the empirical and fitted variograms are plotted in a pdf file
pdf_parameters	list with the fields `test_nr` and `sample_draw`. Additional parameters to name the pdf
metric_manifold	metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot"
model_ts	type of model fitted on the tangent space. It must be chosen among "Intercept", "Coord1", "Coord2", "Additive"
vario_model	type of variogram fitted. It must be chosen among "Gaussian", "Spherical", "Exponential"
distance	type of distance between coordinates. It must be either "Eucldist" or "Geodist"

Value

it returns a list with the following fields

- `resBootstrap`...
- `resAggregated`...
- `model_pred`...

`model_GLS` *Create a GLS model*

Description

Given the coordinates and corresponding manifold values, this function creates a GLS model on the tangent space.

Usage

```
model_GLS(data_manifold, coords, X = NULL, Sigma = NULL,
  metric_manifold = "Frobenius", metric_ts = "Frobenius",
  model_ts = "Additive", vario_model = "Gaussian", n_h = 15,
  distance = "Geodist", max_it = 100, tolerance = 1e-06,
  weight_intrinsic = NULL, tolerance_intrinsic = 1e-06,
  max_sill = NULL, max_a = NULL, param_weighted_vario = NULL,
  plot = FALSE, suppressMes = FALSE, weight_extrinsic = NULL,
  tolerance_map_cor = 1e-06)
```

Arguments

- `data_manifold` list or array [p,p,N] of N symmetric positive definite matrices of dimension p*p
- `coords` N*2 or N*3 matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees
- `X` matrix (N rows and unrestricted number of columns) of additional covariates for the tangent space model, possibly NULL
- `Sigma` p*p matrix representing the tangent point. If NULL the tangent point is computed as the intrinsic mean of `data_manifold`
- `metric_manifold` metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot", "Correlation"
- `metric_ts` metric used on the tangent space. It must be chosen among "Frobenius", "FrobeniusScaled", "Correlation"
- `model_ts` type of model fitted on the tangent space. It must be chosen among "Intercept", "Coord1", "Coord2", "Additive"
- `vario_model` type of variogram fitted. It must be chosen among "Gaussian", "Spherical", "Exponential"
- `n_h` number of bins in the emprical variogram
- `distance` type of distance between coordinates. It must be either "Eucldist" or "Geodist"
- `max_it` max number of iterations for the main loop
- `tolerance` tolerance for the main loop
- `weight_intrinsic` vector of length N to weight the locations in the computation of the intrinsic mean. If NULL a vector of ones is used. Not needed if Sigma is provided
- `tolerance_intrinsic` tolerance for the computation of the intrinsic mean. Not needed if Sigma is provided
- `max_sill` max value allowed for `sill` in the fitted variogram. If NULL it is defined as `1.15*max(emp_vario_values)`
- `max_a` maximum value for `a` in the fitted variogram. If NULL it is defined as `1.15*h_max`
- `param_weighted_vario` List of 7 elements to be provided to consider Kernel weights for the variogram: `weight_vario` (vector of length `N_tot` to weight the locations in the computation of the empirical variogram), `distance_matrix_tot` (`N_tot*N_tot` matrix of distances between the locations), `data_manifold_tot` (list or array `[p,p,N_tot]` of `N_tot` symmetric positive definite matrices of dimension `p*p`),

coords_tot (N_tot*2 or N_tot*3 matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees), X_tot (matrix with N_tot rows and unrestricted number of columns, of additional covariates for the tangent space model. Possibly NULL), h_max (maximum value of distance for which the variogram is computed) indexes_model (indexes corresponding to coords in coords_tot).
Required only in the case metric_manifold = "Correlation"

plot
: boolean. If TRUE the empirical and fitted variograms are plotted

suppressMes
: boolean. If TRUE warning messagges are not printed

weight_extrinsic
: vector of length N to weight the locations in the computation of the extrinsic mean. If NULL weight_intrinsic are used. Needed only if Sigma is not provided and metric_manifold== "Correlation"

tolerance_map_cor
: tolerance to use in the maps.
Required only if metric_manifold== "Correlation"

data_dist_mat
: Matrix of dimension N*N of distances between data points. If not provided it is computed using distance

Details

The manifold values are mapped on the tangent space and then a GLS model is fitted to them. A first estimate of the beta coefficients is obtained assuming spatially uncorrelated errors. Then, in the main the loop, new estimates of the beta are obtained as a result of a weighted least square problem where the weight matrix is the inverse of gamma_matrix. The residuals (residuals = data_ts - fitted) are updated accordingly. The parameters of the variogram fitted to the residuals (and used in the evaluation of the gamma_matrix) are computed using Gauss-Newton with backtrack method to solve the associated non-linear least square problem. The stopping criteria is based on the absolute value of the variogram residuals' norm if ker.width.vario=0, while it is based on its increment otherwise.

Value

A list with the following fields:

beta
: vector of the beta matrices of the fitted model

gamma_matrix
: N*N covariogram matrix

residuals
: vector of the N residual matrices

emp_vario_values
: vector of empircal variogram values in correspondence of h_vec

h_vec
: vector of positions at which the empirical variogram is computed

fitted_par_vario
: estimates of *nugget*, *sill-nugget* and *practical range*

iterations
: number of iterations of the main loop

Sigma
: tangent point

References

D. Pigoli, A. Menafoglio & P. Secchi (2016): Kriging prediction for manifold-valued random fields. Journal of Multivariate Analysis, 145, 117-131.

Examples

```
data_manifold_model <- Manifoldgstat::rCov
coords_model <- Manifoldgstat::rGrid
Sigma <- matrix(c(2,1,1,1), 2,2)
model = model_GLS(data_manifold = data_manifold_model, coords = coords_model, Sigma = Sigma,
            metric_manifold = "Frobenius", metric_ts = "Frobenius", model_ts = "Coord1",
              vario_model = "Spherical", n_h = 15, distance = "Eucldist", max_it = 100,
                tolerance = 1e-7, plot = TRUE)
```

`model_kriging` *Create a GLS model and directly perform kriging*

Description

Given the coordinates and corresponding manifold values, this function firstly creates a GLS model on the tangent space, and then it performs kriging on the new locations.

Usage

```
model_kriging(data_manifold, coords, X = NULL, Sigma = NULL,
  metric_manifold = "Frobenius", metric_ts = "Frobenius",
  model_ts = "Additive", vario_model = "Gaussian", n_h = 15,
  distance = NULL, data_dist_mat = NULL, data_grid_dist_mat = NULL,
  max_it = 100, tolerance = 1e-06, weight_intrinsic = NULL,
  tolerance_intrinsic = 1e-06, max_sill = NULL, max_a = NULL,
  param_weighted_vario = NULL, new_coords, X_new = NULL,
  create_pdf_vario = TRUE, pdf_parameters = NULL,
  suppressMes = FALSE, weight_extrinsic = NULL,
  tolerance_map_cor = 1e-06)
```

Arguments

`data_manifold`	list or array [`p,p,N`] of `N` symmetric positive definite matrices of dimension `p*p`
`coords`	`N*2` or `N*3` matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees
`X`	matrix (N rows and unrestricted number of columns) of additional covariates for the tangent space model, possibly NULL
`Sigma`	`p*p` matrix representing the tangent point. If NULL the tangent point is computed as the intrinsic mean of `data_manifold`
`metric_manifold`	metric used on the manifold. It must be chosen among "Frobenius", "LogEuclidean", "SquareRoot", "Correlation"
`metric_ts`	metric used on the tangent space. It must be chosen among "Frobenius", "FrobeniusScaled", "Correlation"
`model_ts`	type of model fitted on the tangent space. It must be chosen among "Intercept", "Coord1", "Coord2", "Additive"
`vario_model`	type of variogram fitted. It must be chosen among "Gaussian", "Spherical", "Exponential"
`n_h`	number of bins in the emprical variogram

distance
: type of distance between coordinates. It must be either "Eucldist" or "Geodist"

data_dist_mat
: Matrix of dimension N*N of distances between data points. If not provided it is computed using distance

data_grid_dist_mat
: Matrix of dimension N*M of distances between data points and grid points. If not provided it is computed using distance

max_it
: max number of iterations for the main loop

tolerance
: tolerance for the main loop

weight_intrinsic
: vector of length N to weight the locations in the computation of the intrinsic mean. If NULL a vector of ones is used. Not needed if Sigma is provided

tolerance_intrinsic
: tolerance for the computation of the intrinsic mean. Not needed if Sigma is provided

max_sill
: max value allowed for sill in the fitted variogram. If NULL it is defined as 1.15*max(emp_vario_values)

max_a
: maximum value for a in the fitted variogram. If NULL it is defined as 1.15*h_max

param_weighted_vario
: List of 7 elements to be provided to consider Kernel weights for the variogram: weight_vario (vector of length N_tot to weight the locations in the computation of the empirical variogram), distance_matrix_tot (N_tot*N_tot matrix of distances between the locations), data_manifold_tot (list or array [p,p,N_tot] of N_tot symmetric positive definite matrices of dimension p*p, coords_tot (N_tot*2 or N_tot*3 matrix of [lat,long], [x,y] or [x,y,z] coordinates. [lat,long] are supposed to be provided in signed decimal degrees), X_tot (matrix with N_tot rows and unrestricted number of columns, of additional covariates for the tangent space model. Possibly NULL), indexes_model (indexes corresponding to coords in coords_tot)

new_coords
: matrix of coordinates for the new locations where to perform kriging

X_new
: matrix (with the same number of rows of new_coords) of additional covariates for the new locations, possibly NULL

create_pdf_vario
: boolean. If TRUE the empirical and fitted variograms are plotted in a pdf file

pdf_parameters
: list with the fields test_nr and sample_draw. Additional parameters to name the pdf

suppressMes
: boolean. If TRUE warning messagges are not printed

weight_extrinsic
: vector of length N to weight the locations in the computation of the extrinsic mean. If NULL weight_intrinsic are used. Needed only if Sigma is not provided and metric_manifold== "Correlation"

tolerance_map_cor
: tolerance to use in the maps.
Required only if metric_manifold== "Correlation"

Details

The manifold values are mapped on the tangent space and then a GLS model is fitted to them. A first estimate of the beta coefficients is obtained assuming spatially uncorrelated errors. Then, in the main the loop, new estimates of the beta are obtained as a result of a weighted least square problem where the weight matrix is the inverse of gamma_matrix. The residuals (residuals = data_ts - fitted) are updated accordingly. The parameters of the variogram fitted to the residuals (and used in the evaluation of the gamma_matrix) are computed using Gauss-Newton with backtrack method to solve the associated non-linear least square problem. The stopping criteria is based on the absolute value of the variogram residuals' norm if ker.width.vario=0, while it is based on its increment otherwise. Once the model is computed, simple kriging on the tangent space is performed in correspondence of the new locations and eventually the estimates are mapped to the manifold.

Value

list with the following fields:

- beta: vector of the beta matrices of the fitted model
- gamma_matrix: N*N covariogram matrix
- residuals: vector of the N residual matrices
- emp_vario_values: vector of empircal variogram values in correspondence of h_vec
- h_vec: vector of positions at which the empirical variogram is computed
- fitted_par_vario: estimates of *nugget*, *sill-nugget* and *practical range*
- iterations: number of iterations of the main loop
- Sigma: tangent point
- prediction: vector of matrices predicted at the new locations

References

D. Pigoli, A. Menafoglio & P. Secchi (2016): Kriging prediction for manifold-valued random fields. Journal of Multivariate Analysis, 145, 117-131.

Examples

```
data_manifold_tot <- Manifoldgstat::fieldCov
data_manifold_model <- Manifoldgstat::rCov
coords_model <- Manifoldgstat::rGrid
coords_tot <- Manifoldgstat::gridCov
Sigma <- matrix(c(2,1,1,1), 2,2)

result = model_kriging (data_manifold = data_manifold_model, coords = coords_model,
                        Sigma = Sigma, metric_manifold = "Frobenius",
                        metric_ts = "Frobenius", model_ts = "Coord1",
                        vario_model = "Spherical", n_h = 15, distance = "Eucldist",
                        max_it = 100, tolerance = 10e-7, new_coords = coords_model)
result_tot = model_kriging (data_manifold = data_manifold_model, coords = coords_model,
                            metric_ts = "Frobenius", Sigma = Sigma,
                            metric_manifold = "Frobenius", model_ts = "Coord1",
                            vario_model = "Spherical", n_h = 15, distance = "Eucldist",
                            max_it = 100, tolerance = 10e-7, new_coords = coords_tot,
                            create_pdf_vario = FALSE)
x.min=min(coords_tot[,1])
x.max=max(coords_tot[,1])
y.min=min(coords_tot[,2])
y.max=max(coords_tot[,2])
```

```
dimgrid=dim(coords_tot)[1]
radius = 0.02

par(cex=1.25)
plot(0,0, asp=1, col=fields::tim.colors(100), ylim=c(y.min,y.max), xlim=c(x.min, x.max),
      pch='', xlab='', ylab='', main = "Real Values")
for(i in 1:dimgrid){
 if(i %% 3 == 0)
    car::ellipse(c(coords_tot[i,1],coords_tot[i,2]) , data_manifold_tot[,,i],
                                 radius=radius, center.cex=.5, col='navyblue')
}
rect(x.min, y.min, x.max, y.max)

for(i in 1:250)
{ car::ellipse(c(coords_model[i,1],coords_model[i,2]) , data_manifold_model[,,i],
               radius=radius, center.cex=.5, col='green')}
rect(x.min, y.min, x.max, y.max)

par(cex=1.25)
plot(0,0, asp=1, col=fields::tim.colors(100), ylim=c(y.min,y.max),xlim=c(x.min, x.max),
      pch='',  xlab='', ylab='',main = "Predicted values")

for(i in 1:dimgrid){
  if(i %% 3 == 0)
     car::ellipse(c(coords_tot[i,1],coords_tot[i,2]), result_tot$prediction[[i]],
                                 radius=radius, center.cex=.5, col='navyblue' )
}
rect(x.min, y.min, x.max, y.max)

for(i in 1:250)
{ car::ellipse(c(rGrid[i,1],rGrid[i,2]), result$prediction[[i]],radius=radius,
               center.cex=.5, col='red')
}
rect(x.min, y.min, x.max, y.max)
```

Index