Package 'GpGp'

October 16, 2024

Type Package

Title Fast Gaussian Process Computation Using Vecchia's Approximation

Version 0.5.1

Date 2024-10-15

Maintainer Joseph Guinness <joeguinness@gmail.com>

Description Functions for fitting and doing predictions with Gaussian process models using Vecchia's (1988) approximation. Package also includes functions for reordering input locations, finding ordered nearest neighbors (with help from 'FNN' package), grouping operations, and conditional simulations. Covariance functions for spatial and spatial-temporal data on Euclidean domains and spheres are provided. The original approximation is due to Vecchia (1988) <http://www.jstor.org/stable/2345768>, and the reordering and grouping methods are from Guinness (2018) <doi:10.1080/00401706.2018.1437476>. Model fitting employs a Fisher scoring algorithm described in Guinness (2019) <doi:10.48550/arXiv.1905.08374>.

Depends R (>= 2.10)

License MIT + file LICENSE

Imports Rcpp (>= 0.12.13), FNN

Suggests fields, knitr, rmarkdown, testthat, maps

LinkingTo Rcpp, RcppArmadillo, BH

RoxygenNote 7.3.2

LazyData true

NeedsCompilation yes

Author Joseph Guinness [aut, cre], Matthias Katzfuss [aut], Youssef Fahmy [aut]

Repository CRAN

Date/Publication 2024-10-16 04:30:14 UTC

Contents

`argo2016` *Ocean temperatures from Argo profiling floats*

Description

A dataset containing ocean temperature measurements from three pressure levels (depths), measured by profiling floats from the Argo program. Data collected in Jan, Feb, and March of 2016.

Usage

```
argo2016
```

Format

A data frame with 32436 rows and 6 columns

lon longitude in degrees between 0 and 360

lat latitude in degrees between -90 and 90

day time in days

temp100 Temperature at 100 dbars (roughly 100 meters)

temp150 Temperature at 150 dbars (roughly 150 meters)

temp200 Temperature at 200 dbars (roughly 200 meters)

Source

Mikael Kuusela. Argo program: https://argo.ucsd.edu/

`condition_number` *compute condition number of matrix*

Description

compute condition number of matrix

Usage

```
condition_number(info)
```

Arguments

info	matrix

`cond_sim` *Conditional Simulation using Vecchia's approximation*

Description

With the prediction locations ordered after the observation locations, an approximation for the inverse Cholesky of the covariance matrix is computed, and standard formulas are applied to obtain a conditional simulation.

Usage

```
cond_sim(
  fit = NULL,
  locs_pred,
  X_pred,
  y_obs = fit$y,
  locs_obs = fit$locs,
  X_obs = fit$X,
  beta = fit$betahat,
  covparms = fit$covparms,
  covfun_name = fit$covfun_name,
  m = 60,
```

```
    reorder = TRUE,
    st_scale = NULL,
    nsims = 1
)
```

Arguments

`fit`	GpGp_fit object, the result of `fit_model`
`locs_pred`	prediction locations
`X_pred`	Design matrix for predictions
`y_obs`	Observations associated with locs_obs
`locs_obs`	observation locations
`X_obs`	Design matrix for observations
`beta`	Linear mean parameters
`covparms`	Covariance parameters
`covfun_name`	Name of covariance function
`m`	Number of nearest neighbors to use. Larger `m` gives better approximations.
`reorder`	TRUE/FALSE for whether reordering should be done. This should generally be kept at TRUE, unless testing out the effect of reordering.
`st_scale`	amount by which to scale the spatial and temporal dimensions for the purpose of selecting neighbors. We recommend setting this manually when using a spatial-temporal covariance function. When `lonlat = TRUE`, spatial scale is in radians (earth radius = 1).
`nsims`	Number of conditional simulations to return.

Details

We can specify either a GpGp_fit object (the result of `fit_model`), OR manually enter the covariance function and parameters, the observations, observation locations, and design matrix. We must specify the prediction locations and the prediction design matrix.

`expit` *expit function and integral of expit function*

Description

expit function and integral of expit function

Usage

```
expit(x)

intexpit(x)
```

Arguments

`x`	argument to expit or intexpit function

---

exponential_anisotropic2D

Geometrically anisotropic exponential covariance function (two dimensions)

Description

From a matrix of locations and covariance parameters of the form (variance, L11, L21, L22, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_anisotropic2D(covparms, locs)

d_exponential_anisotropic2D(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, L11, L21, L22, nugget)
`locs`	A matrix with `n` rows and `2` columns. Each row of locs is a point in R^2.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_exponential_anisotropic2D()`: Derivatives of anisotropic exponential covariance

Parameterization

The covariance parameter vector is (variance, L11, L21, L22, nugget) where L11, L21, L22, are the three non-zero entries of a lower-triangular matrix L. The covariances are

$$M(x,y) = \sigma^2 exp(-||Lx - Ly||)$$

This means that L11 is interpreted as an inverse range parameter in the first dimension. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

exponential_anisotropic3D

Geometrically anisotropic exponential covariance function (three dimensions)

Description

From a matrix of locations and covariance parameters of the form (variance, L11, L21, L22, L31, L32, L33, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_anisotropic3D(covparms, locs)

d_exponential_anisotropic3D(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, L11, L21, L22, L31, L32, L33, nugget)
`locs`	A matrix with `n` rows and `3` columns. Each row of locs is a point in R^3.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_exponential_anisotropic3D()`: Derivatives of anisotropic exponential covariance

Parameterization

The covariance parameter vector is (variance, L11, L21, L22, L31, L32, L33, nugget) where L11, L21, L22, L31, L32, L33 are the six non-zero entries of a lower-triangular matrix L. The covariances are

$$M(x, y) = \sigma^2 exp(-||Lx - Ly||)$$

This means that L11 is interpreted as an inverse range parameter in the first dimension. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

exponential_anisotropic3D_alt

Geometrically anisotropic exponential covariance function (three dimensions, alternate parameterization)

Description

From a matrix of locations and covariance parameters of the form (variance, B11, B12, B13, B22, B23, B33, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_anisotropic3D_alt(covparms, locs)

d_exponential_anisotropic3D_alt(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, B11, B12, B13, B22, B23, B33, smoothness, nugget)
`locs`	A matrix with `n` rows and `3` columns. Each row of locs is a point in R^3.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_exponential_anisotropic3D_alt()`: Derivatives of anisotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, B11, B12, B13, B22, B23, B33, smoothness, nugget) where B11, B12, B13, B22, B23, B33, transform the three coordinates as

$$u_1 = B11[x_1 + B12x_2 + (B13 + B12B23)x_3]$$

$$u_2 = B22[x_2 + B23x_3]$$

$$u_3 = B33[x_3]$$

(B13,B23) can be interpreted as a drift vector in space over time if first two dimensions are space and third is time. Assuming x is transformed to u and y transformed to v, the covariances are

$$M(x, y) = \sigma^2 exp(-||u - v||)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

exponential_isotropic *Isotropic exponential covariance function*

Description

From a matrix of locations and covariance parameters of the form (variance, range, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_isotropic(covparms, locs)

d_exponential_isotropic(covparms, locs)

d_matern15_isotropic(covparms, locs)

d_matern25_isotropic(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range, nugget)
locs	A matrix with n rows and d columns. Each row of locs is a point in R^d.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_exponential_isotropic(): Derivatives of isotropic exponential covariance
- d_matern15_isotropic(): Derivatives of isotropic matern covariance with smoothness 1.5
- d_matern25_isotropic(): Derivatives of isotropic matern covariance function with smoothness 2.5

Parameterization

The covariance parameter vector is (variance, range, nugget) = $(\sigma^2, \alpha, \tau^2)$, and the covariance function is parameterized as

$$M(x, y) = \sigma^2 exp(-||x - y||/\alpha)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

exponential_nonstat_var
Isotropic exponential covariance function, nonstationary variances

Description

From a matrix of locations and covariance parameters of the form (variance, range, nugget, <nonstat variance parameters>), return the square matrix of all pairwise covariances.

Usage

```
exponential_nonstat_var(covparms, Z)

d_exponential_nonstat_var(covparms, Z)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range, nugget, <nonstat variance parameters>). The number of nonstationary variance parameters should equal p.
Z	A matrix with n rows and 2 columns for spatial locations + p columns describing spatial basis functions. Each row of locs gives a point in R^2 (two dimensions only!) + the value of p spatial basis functions.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_exponential_nonstat_var(): Derivatives with respect to parameters

Parameterization

This covariance function multiplies the isotropic exponential covariance by a nonstationary variance function. The form of the covariance is

$$C(x, y) = exp(\phi(x) + \phi(y))M(x, y)$$

where M(x,y) is the isotropic exponential covariance, and

$$\phi(x) = c_1\phi_1(x) + ... + c_p\phi_p(x)$$

where $\phi_1, ..., \phi_p$ are the spatial basis functions contained in the last p columns of Z, and $c_1, ..., c_p$ are the nonstationary variance parameters.

`exponential_scaledim` *Exponential covariance function, different range parameter for each dimension*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, ..., range_d, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_scaledim(covparms, locs)

d_exponential_scaledim(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, ..., range_d, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_exponential_scaledim()`: Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, ..., range_d, nugget). The covariance function is parameterized as

$$M(x, y) = \sigma^2 exp(-||D^{-1}(x - y)||)$$

where D is a diagonal matrix with (range_1, ..., range_d) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`exponential_spacetime` *Spatial-Temporal exponential covariance function*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, range_2, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_spacetime(covparms, locs)

d_exponential_spacetime(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, range_2, nugget). range_1 is the spatial range, and range_2 is the temporal range.
`locs`	A matrix with `n` rows and `d+1` columns. Each row of locs is a point in R^(d+1). The first `d` columns should contain the spatial coordinates. The last column contains the times.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_exponential_spacetime()`: Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, range_2, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2 exp(-||D^{-1}(x-y)||)$$

where D is a diagonal matrix with (range_1, ..., range_1, range_2) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

exponential_sphere *Isotropic exponential covariance function on sphere*

Description

From a matrix of longitudes and latitudes and a vector covariance parameters of the form (variance, range, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_sphere(covparms, lonlat)

d_exponential_sphere(covparms, lonlat)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, nugget). Range parameter assumes that the sphere has radius 1 (units are radians).
`lonlat`	A matrix with `n` rows and one column with longitudes in (-180,180) and one column of latitudes in (-90,90). Each row of lonlat describes a point on the sphere.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlat[i,]` and `lonlat[j,]`.

Functions

- `d_exponential_sphere()`: Derivatives with respect to parameters

Covariances on spheres

The function first calculates the (x,y,z) 3D coordinates, and then inputs the resulting locations into `exponential_isotropic`. This means that we construct covariances on the sphere by embedding the sphere in a 3D space. There has been some concern expressed in the literature that such embeddings may produce distortions. The source and nature of such distortions has never been articulated, and to date, no such distortions have been documented. Guinness and Fuentes (2016) argue that 3D embeddings produce reasonable models for data on spheres.

`exponential_spheretime` *Exponential covariance function on sphere x time*

Description

From a matrix of longitudes, latitudes, and times, and a vector covariance parameters of the form (variance, range_1, range_2, nugget), return the square matrix of all pairwise covariances.

Usage

```
exponential_spheretime(covparms, lonlattime)

d_exponential_spheretime(covparms, lonlattime)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, range_2, nugget), where range_1 is a spatial range (assuming sphere of radius 1), and range_2 is a temporal range.
`lonlattime`	A matrix with `n` rows and three columns: longitudes in (-180,180), latitudes in (-90,90), and times. Each row of lonlattime describes a point on the sphere x time.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlattime[i,]` and `lonlattime[j,]`.

Functions

- `d_exponential_spheretime()`: Derivatives with respect to parameters.

Covariances on spheres

The function first calculates the (x,y,z) 3D coordinates, and then inputs the resulting locations into `exponential_spacetime`. This means that we construct covariances on the sphere by embedding the sphere in a 3D space. There has been some concern expressed in the literature that such embeddings may produce distortions. The source and nature of such distortions has never been articulated, and to date, no such distortions have been documented. Guinness and Fuentes (2016) argue that 3D embeddings produce reasonable models for data on spheres.

`exponential_spheretime_warp` *Deformed exponential covariance function on sphere*

Description

From a matrix of longitudes, latitudes, times, and a vector covariance parameters of the form (variance, range_1, range_2, nugget, <5 warping parameters>), return the square matrix of all pairwise covariances.

Usage

```
exponential_spheretime_warp(covparms, lonlattime)

d_exponential_spheretime_warp(covparms, lonlattime)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, range_2, nugget, <5 warping parameters>). range_1 is a spatial range parameter that assumes that the sphere has radius 1 (units are radians). range_2 is a temporal range parameter.
`lonlattime`	A matrix with `n` rows and three columns: longitudes in (-180,180), latitudes in (-90,90), and times. Each row of lonlattime describes a point on the sphere x time.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlat[i,]` and `lonlat[j,]`.

Functions

- `d_exponential_spheretime_warp()`: Derivatives with respect to parameters

Warpings

The function first calculates the (x,y,z) 3D coordinates, and then "warps" the locations to $(x, y, z) + \Phi(x, y, z)$, where Φ is a warping function composed of gradients of spherical harmonic functions of degree 2. See Guinness (2019, "Gaussian Process Learning via Fisher Scoring of Vecchia's Approximation") for details. The warped locations are input into `exponential_spacetime`. The function does not do temporal warping.

`exponential_sphere_warp` *Deformed exponential covariance function on sphere*

Description

From a matrix of longitudes and latitudes and a vector covariance parameters of the form (variance, range, nugget, <5 warping parameters>), return the square matrix of all pairwise covariances.

Usage

```
exponential_sphere_warp(covparms, lonlat)

d_exponential_sphere_warp(covparms, lonlat)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, nugget, <5 warping parameters>). Range parameter assumes that the sphere has radius 1 (units are radians).
`lonlat`	A matrix with `n` rows and one column with longitudes in (-180,180) and one column of latitudes in (-90,90). Each row of lonlat describes a point on the sphere.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlat[i,]` and `lonlat[j,]`.

Functions

- `d_exponential_sphere_warp()`: Derivatives with respect to parameters

Warpings

The function first calculates the (x,y,z) 3D coordinates, and then "warps" the locations to $(x, y, z) + \Phi(x, y, z)$, where Φ is a warping function composed of gradients of spherical harmonic functions of degree 2. See Guinness (2019, "Gaussian Process Learning via Fisher Scoring of Vecchia's Approximation") for details. The warped locations are input into `exponential_isotropic`.

fast_Gp_sim *Approximate GP simulation*

Description

Calculates an approximation to the inverse Cholesky factor of the covariance matrix using Vecchia's approximation, then the simulation is produced by solving a linear system with a vector of uncorrelated standard normals

Usage

```
fast_Gp_sim(covparms, covfun_name = "matern_isotropic", locs, m = 30)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
m	Number of nearest neighbors to use in approximation

Value

vector of simulated values

Examples

```
locs <- as.matrix( expand.grid( (1:50)/50, (1:50)/50 ) )
y <- fast_Gp_sim(c(4,0.2,0.5,0), "matern_isotropic",  locs, 30 )
fields::image.plot( matrix(y,50,50) )
```

fast_Gp_sim_Linv *Approximate GP simulation with specified Linverse*

Description

In situations where we want to do many gaussian process simulations from the same model, we can compute Linverse once and reuse it, rather than recomputing for each identical simulation. This function also allows the user to input the vector of standard normals z.

Usage

```
fast_Gp_sim_Linv(Linv, NNarray, z = NULL)
```

Arguments

`Linv`	Matrix containing the entries of Linverse, usually the output from `vecchia_Linv`.
`NNarray`	Matrix of nearest neighbor indices, usually the output from `find_ordered_nn`
`z`	Optional vector of standard normals. If not specified, these are computed within the function.

Value

vector of simulated values

Examples

```
locs <- as.matrix( expand.grid( (1:50)/50, (1:50)/50 ) )
ord <- order_maxmin(locs)
locsord <- locs[ord,]
m <- 10
NNarray <- find_ordered_nn(locsord,m)
covparms <- c(2, 0.2, 1, 0)
Linv <- vecchia_Linv( covparms, "matern_isotropic", locsord, NNarray )
y <- fast_Gp_sim_Linv(Linv,NNarray)
y[ord] <- y
fields::image.plot( matrix(y,50,50) )
```

`find_ordered_nn` *Find ordered nearest neighbors.*

Description

Given a matrix of locations, find the `m` nearest neighbors to each location, subject to the neighbors coming previously in the ordering. The algorithm uses the kdtree algorithm in the FNN package, adapted to the setting where the nearest neighbors must come from previous in the ordering.

Usage

```
find_ordered_nn(locs, m, lonlat = FALSE, st_scale = NULL)
```

Arguments

`locs`	A matrix of locations. Each row of `locs` contains a location, which can be a point in Euclidean space R^d, a point in space-time R^d x T, a longitude and latitude (in degrees) giving a point on the sphere, or a longitude, latitude, and time giving a point in the sphere-time domain.
`m`	Number of neighbors to return
`lonlat`	TRUE/FALSE whether locations are longitudes and latitudes.

st_scale — factor by which to scale the spatial and temporal coordinates for distance calculations. The function assumes that the last column of the locations is the temporal dimension, and the rest of the columns are spatial dimensions. The spatial dimensions are divided by st_scale[1], and the temporal dimension is divided by st_scale[2], before distances are calculated. If st_scale is NULL, no scaling is used. We recommend setting st_scale manually so that each observation gets neighbors that hail multiple directions in space and time.

Value

An matrix containing the indices of the neighbors. Row i of the returned matrix contains the indices of the nearest m locations to the i'th location. Indices are ordered within a row to be increasing in distance. By convention, we consider a location to neighbor itself, so the first entry of row i is i, the second entry is the index of the nearest location, and so on. Because each location neighbors itself, the returned matrix has m+1 columns.

Examples

```
locs <- as.matrix( expand.grid( (1:40)/40, (1:40)/40 ) )
ord <- order_maxmin(locs)        # calculate an ordering
locsord <- locs[ord,]            # reorder locations
m <- 20
NNarray <- find_ordered_nn(locsord,20)  # find ordered nearest 20 neighbors
ind <- 100
# plot all locations in gray, first ind locations in black,
# ind location with magenta circle, m neighhbors with blue circle
plot( locs[,1], locs[,2], pch = 16, col = "gray" )
points( locsord[1:ind,1], locsord[1:ind,2], pch = 16 )
points( locsord[ind,1], locsord[ind,2], col = "magenta", cex = 1.5 )
points( locsord[NNarray[ind,2:(m+1)],1],
    locsord[NNarray[ind,2:(m+1)],2], col = "blue", cex = 1.5 )
```

---

find_ordered_nn_brute *Naive brute force nearest neighbor finder*

Description

Naive brute force nearest neighbor finder

Usage

```
find_ordered_nn_brute(locs, m)
```

Arguments

locs	matrix of locations
m	number of neighbors

Value

An matrix containing the indices of the neighbors. Row i of the returned matrix contains the indices of the nearest m locations to the i'th location. Indices are ordered within a row to be increasing in distance. By convention, we consider a location to neighbor itself, so the first entry of row i is i, the second entry is the index of the nearest location, and so on. Because each location neighbors itself, the returned matrix has m+1 columns.

---

fisher_scoring *Fisher scoring algorithm*

Description

Fisher scoring algorithm

Usage

```
fisher_scoring(
  likfun,
  start_parms,
  link,
  silent = FALSE,
  convtol = 1e-04,
  max_iter = 40
)
```

Arguments

likfun	likelihood function, returns likelihood, gradient, and hessian
start_parms	starting values of parameters
link	link function for parameters (used for printing)
silent	TRUE/FALSE for suppressing output
convtol	convergence tolerance on step dot grad
max_iter	maximum number of Fisher scoring iterations

fit_model *Estimate mean and covariance parameters*

Description

Given a response, set of locations, (optionally) a design matrix, and a specified covariance function, return the maximum Vecchia likelihood estimates, obtained with a Fisher scoring algorithm.

Usage

```
fit_model(
  y,
  locs,
  X = NULL,
  covfun_name = "matern_isotropic",
  NNarray = NULL,
  start_parms = NULL,
  reorder = TRUE,
  group = TRUE,
  m_seq = c(10, 30),
  max_iter = 40,
  fixed_parms = NULL,
  silent = FALSE,
  st_scale = NULL,
  convtol = 1e-04
)
```

Arguments

y	response vector
locs	matrix of locations. Each row is a single spatial or spatial-temporal location. If using one of the covariance functions for data on a sphere, the first column should be longitudes (-180,180) and the second column should be latitudes (-90,90). If using a spatial-temporal covariance function, the last column should contain the times.
X	design matrix. Each row contains covariates for the corresponding observation in `y`. If not specified, the function sets `X` to be a matrix with a single column of ones, that is, a constant mean function.
covfun_name	string name of a covariance function. See `GpGp` for information about supported covariance funtions.
NNarray	Optionally specified array of nearest neighbor indices, usually from the output of `find_ordered_nn`. If `NULL`, fit_model will compute the nearest neighbors. We recommend that the user not specify this unless there is a good reason to (e.g. if doing a comparison study where one wants to control `NNarray` across different approximations).

start_parms: Optionally specified starting values for parameters. If NULL, fit_model will select default starting values.

reorder: TRUE/FALSE indicating whether maxmin ordering should be used (TRUE) or whether no reordering should be done before fitting (FALSE). If you want to use a customized reordering, then manually reorder y, locs, and X, and then set reorder to FALSE. A random reordering is used when nrow(locs) > 1e5.

group: TRUE/FALSE for whether to use the grouped version of the approximation (Guinness, 2018) or not. The grouped version is used by default and is always recommended.

m_seq: Sequence of values for number of neighbors. By default, a 10-neighbor approximation is maximized, then a 30-neighbor approximation is maximized using the 10 neighbor estimates as starting values. However, one can specify any sequence of numbers of neighbors, e.g. m_seq = c(10,30,60,90).

max_iter: maximum number of Fisher scoring iterations

fixed_parms: Indices of covariance parameters you would like to fix at specific values. If you decide to fix any parameters, you must specify their values in start_parms, along with the starting values for all other parameters. For example, to fix the nugget at zero in exponential_isotropic, set fixed_parms to c(3), and set start_parms to c(4.7,3.1,0). The last element of start_parms (the nugget parameter) is set to zero, while the starting values for the other two parameters are 4.7 and 3.1.

silent: TRUE/FALSE for whether to print some information during fitting.

st_scale: Scaling for spatial and temporal ranges. Only applicable for spatial-temporal models, where it is used in distance calculations when selecting neighbors. st_scale must be specified when covfun_name is a spatial-temporal covariance. See Argo vignette for an example.

convtol: Tolerance for exiting the optimization. Fisher scoring is stopped when the dot product between the step and the gradient is less than convtol.

Details

fit_model is a user-friendly model fitting function that automatically performs many of the auxiliary tasks needed for using Vecchia's approximation, including reordering, computing nearest neighbors, grouping, and optimization. The likelihoods use a small penalty on small nuggets, large spatial variances, and small smoothness parameter.

The Jason-3 windspeed vignette and the Argo temperature vignette are useful sources for a use-cases of the fit_model function for data on sphere. The example below shows a very small example with a simulated dataset in 2d.

Value

An object of class GpGp_fit, which is a list containing covariance parameter estimates, regression coefficients, covariance matrix for mean parameter estimates, as well as some other information relevant to the model fit.

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
covparms <- c(2,0.1,1/2,0)
y <- 7 + fast_Gp_sim(covparms, "matern_isotropic", locs)
X <- as.matrix( rep(1,n) )
## not run
# fit <- fit_model(y, locs, X, "matern_isotropic")
# fit
```

---

get_linkfun — *get link function, whether locations are lonlat and space time*

Description

get link function, whether locations are lonlat and space time

Usage

```
get_linkfun(covfun_name)
```

Arguments

covfun_name	string name of covariance function

---

get_penalty — *get penalty function*

Description

get penalty function

Usage

```
get_penalty(y, X, locs, covfun_name)
```

Arguments

y	response
X	design matrix
locs	locations
covfun_name	string name of covariance function

get_start_parms — *get default starting values of covariance parameters*

Description

get default starting values of covariance parameters

Usage

```
get_start_parms(y, X, locs, covfun_name)
```

Arguments

y	response
X	design matrix
locs	locations
covfun_name	string name of covariance function

GpGp — *GpGp: Fast Gaussian Process Computing.*

Description

Vecchia's (1988) Gaussian process approximation has emerged among its competitors as a leader in computational scalability and accuracy. This package includes implementations of the original approximation, as well as several updates to it, including the reordered and grouped versions of the approximation outlined in Guinness (2018) and the Fisher scoring algorithm described in Guinness (2019). The package supports spatial models, spatial-temporal models, models on spheres, and some nonstationary models.

Details

The main functions of the package are `fit_model`, and `predictions`. `fit_model` returns estimates of covariance parameters and linear mean parameters. The user is expected to select a covariance function and specify it with a string. Currently supported covariance functions are

- `matern_isotropic`
- `exponential_isotropic`
- `matern_anisotropic2D`
- `exponential_anisotropic2D`
- `matern_anisotropic3D`
- `exponential_anisotropic3D`
- `matern_anisotropic3D_alt`

- matern15_isotropic
- matern25_isotropic
- matern35_isotropic
- matern45_isotropic
- matern_scaledim
- exponential_scaledim
- matern15_scaledim
- matern25_scaledim
- matern35_scaledim
- matern45_scaledim
- matern_spacetime
- exponential_spacetime
- matern_nonstat_var
- exponential_nonstat_var
- matern_sphere
- exponential_sphere
- matern_spheretime
- exponential_spheretime
- matern_sphere_warp
- exponential_sphere_warp
- matern_spheretime_warp
- exponential_spheretime_warp

If there are covariates, they can be expressed via a design matrix `X`, each row containing the covariates corresponding to the same row in `locs`.

For predictions, the user should specify prediction locations `locs_pred` and a prediction design matrix `X_pred`.

The vignettes are intended to be helpful for getting a sense of how these functions work.

For Gaussian process researchers, the package also provides access to functions for computing the likelihood, gradient, and Fisher information with respect to covariance parameters; reordering functions, nearest neighbor-finding functions, grouping (partitioning) functions, and approximate simulation functions.

Author(s)

Maintainer: Joseph Guinness <joeguinness@gmail.com>

Authors:

- Matthias Katzfuss <katzfuss@gmail.com>
- Youssef Fahmy <yf297@cornell.edu>

group_obs *Automatic grouping (partitioning) of locations*

Description

Take in an array of nearest neighbors, and automatically partition the array into groups that share neighbors. This is helpful to speed the computations and improve their accuracy. The function returns a list, with each list element containing one or several rows of NNarray. The algorithm attempts to find groupings such that observations within a group share many common neighbors.

Usage

```
group_obs(NNarray, exponent = 2)
```

Arguments

NNarray	Matrix of nearest neighbor indices, usually the result of find_ordered_nn.
exponent	Within the algorithm, two groups are merged if the number of unique neighbors raised to the exponent power is less than the sum of the unique numbers raised to the exponent power from the two groups.

Value

A list with elements defining the grouping. The list entries are:

- all_inds: vector of all indices of all blocks.
- last_ind_of_block: The ith entry tells us the location in all_inds of the last index of the ith block. Thus the length of last_ind_of_block is the number of blocks, and last_ind_of_block can be used to chop all_inds up into blocks.
- global_resp_inds: The ith entry tells us the index of the ith response, as ordered in all_inds.
- local_resp_inds: The ith entry tells us the location within the block of the response index.
- last_resp_of_block: The ith entry tells us the location within local_resp_inds and global_resp_inds of the last index of the ith block. last_resp_of_block is to global_resp_inds and local_resp_inds as last_ind_of_block is to all_inds.

Examples

```
locs <- matrix( runif(200), 100, 2 )   # generate random locations
ord <- order_maxmin(locs)              # calculate an ordering
locsord <- locs[ord,]                  # reorder locations
m <- 10
NNarray <- find_ordered_nn(locsord,m)  # m nearest neighbor indices
NNlist2 <- group_obs(NNarray)          # join blocks if joining reduces squares
NNlist3 <- group_obs(NNarray,3)        # join blocks if joining reduces cubes
object.size(NNarray)
object.size(NNlist2)
```

```
object.size(NNlist3)
mean( NNlist2[["local_resp_inds"]] - 1 )   # average number of neighbors (exponent 2)
mean( NNlist3[["local_resp_inds"]] - 1 )   # average number of neighbors (exponent 3)

all_inds <- NNlist2$all_inds
last_ind_of_block <- NNlist2$last_ind_of_block
inds_of_block_2 <- all_inds[ (last_ind_of_block[1] + 1):last_ind_of_block[2] ]

local_resp_inds <- NNlist2$local_resp_inds
global_resp_inds <- NNlist2$global_resp_inds
last_resp_of_block <- NNlist2$last_resp_of_block
local_resp_of_block_2 <-
    local_resp_inds[(last_resp_of_block[1]+1):last_resp_of_block[2]]

global_resp_of_block_2 <-
    global_resp_inds[(last_resp_of_block[1]+1):last_resp_of_block[2]]
inds_of_block_2[local_resp_of_block_2]
# these last two should be the same
```

jason3 *Windspeed measurements from Jason-3 Satellite*

Description

A dataset containing lightly preprocessed windspeed values from the Jason-3 satellite. Observations near clouds and ice have been removed, and the data have been aggregated (averaged) over 10 second intervals. Jason-3 reports windspeeds over the ocean only. The data are from a six day period between August 4 and 9 of 2016.

Usage

```
jason3
```

Format

A data frame with 18973 rows and 4 columns

windspeed wind speed, in maters per second

lon longitude in degrees between 0 and 360

lat latitude in degrees between -90 and 90

time time in seconds from midnight August 4

Source

https://www.ncei.noaa.gov/products/jason-satellite-products

Linv_mult — *Multiply approximate inverse Cholesky by a vector*

Description

Vecchia's approximation implies a sparse approximation to the inverse Cholesky factor of the covariance matrix. This function returns the result of multiplying that matrix by a vector.

Usage

```
Linv_mult(Linv, z, NNarray)
```

Arguments

Linv	Entries of the sparse inverse Cholesky factor, usually the output from vecchia_Linv.
z	the vector to be multiplied
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

the product of the sparse inverse Cholesky factor with a vector

Examples

```
n <- 2000
locs <- matrix( runif(2*n), n, 2 )
covparms <- c(2, 0.2, 0.75, 0.1)
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
Linv <- vecchia_Linv( covparms, "matern_isotropic", locs, NNarray )
z1 <- rnorm(n)
y <- fast_Gp_sim_Linv(Linv,NNarray,z1)
z2 <- Linv_mult(Linv, y, NNarray)
print( sum( (z1-z2)^2 ) )
```

Linv_t_mult — *Multiply transpose of approximate inverse Cholesky by a vector*

Description

Vecchia's approximation implies a sparse approximation to the inverse Cholesky factor of the covariance matrix. This function returns the result of multiplying the transpose of that matrix by a vector.

Usage

```
Linv_t_mult(Linv, z, NNarray)
```

Arguments

Linv	Entries of the sparse inverse Cholesky factor, usually the output from vecchia_Linv.
z	the vector to be multiplied
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

the product of the transpose of the sparse inverse Cholesky factor with a vector

Examples

```
n <- 2000
locs <- matrix( runif(2*n), n, 2 )
covparms <- c(2, 0.2, 0.75, 0.1)
NNarray <- find_ordered_nn(locs,20)
Linv <- vecchia_Linv( covparms, "matern_isotropic", locs, NNarray )
z1 <- rnorm(n)
z2 <- Linv_t_mult(Linv, z1, NNarray)
```

L_mult *Multiply approximate Cholesky by a vector*

Description

Vecchia's approximation implies a sparse approximation to the inverse Cholesky factor of the covariance matrix. This function returns the result of multiplying the inverse of that matrix by a vector (i.e. an approximation to the Cholesky factor).

Usage

```
L_mult(Linv, z, NNarray)
```

Arguments

Linv	Entries of the sparse inverse Cholesky factor, usually the output from vecchia_Linv.
z	the vector to be multiplied
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

the product of the Cholesky factor with a vector

Examples

```
n <- 2000
locs <- matrix( runif(2*n), n, 2 )
covparms <- c(2, 0.2, 0.75, 0.1)
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
Linv <- vecchia_Linv( covparms, "matern_isotropic", locs, NNarray )
z <- rnorm(n)
y1 <- fast_Gp_sim_Linv(Linv,NNarray,z)
y2 <- L_mult(Linv, z, NNarray)
print( sum( (y1-y2)^2 ) )
```

L_t_mult *Multiply transpose of approximate Cholesky by a vector*

Description

Vecchia's approximation implies a sparse approximation to the inverse Cholesky factor of the covariance matrix. This function returns the result of multiplying the transpose of the inverse of that matrix by a vector (i.e. an approximation to the transpose of the Cholesky factor).

Usage

```
L_t_mult(Linv, z, NNarray)
```

Arguments

Linv	Entries of the sparse inverse Cholesky factor, usually the output from vecchia_Linv.
z	the vector to be multiplied
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

the product of the transpose of the Cholesky factor with a vector

Examples

```
n <- 2000
locs <- matrix( runif(2*n), n, 2 )
covparms <- c(2, 0.2, 0.75, 0.1)
NNarray <- find_ordered_nn(locs,20)
Linv <- vecchia_Linv( covparms, "matern_isotropic", locs, NNarray )
z1 <- rnorm(n)
z2 <- L_t_mult(Linv, z1, NNarray)
```

---

matern15_isotropic *Isotropic Matern covariance function, smoothness = 1.5*

Description

From a matrix of locations and covariance parameters of the form (variance, range, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern15_isotropic(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Parameterization

The covariance parameter vector is (variance, range, nugget) = $(\sigma^2, \alpha, \tau^2)$, and the covariance function is parameterized as

$$M(x, y) = \sigma^2(1 + ||x - y||)exp(-||x - y||/\alpha)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern15_scaledim *Matern covariance function, smoothess = 1.5, different range parameter for each dimension*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, ..., range_d, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern15_scaledim(covparms, locs)

d_matern15_scaledim(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range_1, ..., range_d, nugget)
locs	A matrix with n rows and d columns. Each row of locs is a point in R^d.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern15_scaledim()`: Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, ..., range_d, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2(1 + ||D^{-1}(x-y)||)exp(-||D^{-1}(x-y)||)$$

where D is a diagonal matrix with (range_1, ..., range_d) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`matern25_isotropic` *Isotropic Matern covariance function, smoothness = 2.5*

Description

From a matrix of locations and covariance parameters of the form (variance, range, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern25_isotropic(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Parameterization

The covariance parameter vector is (variance, range, nugget) = $(\sigma^2, \alpha, \tau^2)$, and the covariance function is parameterized as

$$M(x, y) = \sigma^2(1 + ||x - y||/\alpha + ||x - y||^2/3\alpha^2)exp(-||x - y||/\alpha)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`matern25_scaledim` *Matern covariance function, smoothess = 2.5, different range parameter for each dimension*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, ..., range_d, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern25_scaledim(covparms, locs)

d_matern25_scaledim(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range_1, ..., range_d, nugget)
locs	A matrix with n rows and d columns. Each row of locs is a point in R^d.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_matern25_scaledim(): Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, ..., range_d, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2(1 + ||D^{-1}(x-y)|| + ||D^{-1}(x-y)||^2/3.0)exp(-||D^{-1}(x-y)||)$$

where D is a diagonal matrix with (range_1, ..., range_d) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

---

matern35_isotropic *Isotropic Matern covariance function, smoothness = 3.5*

Description

From a matrix of locations and covariance parameters of the form (variance, range, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern35_isotropic(covparms, locs)

d_matern35_isotropic(covparms, locs)

d_matern45_isotropic(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range, nugget)
locs	A matrix with n rows and d columns. Each row of locs is a point in R^d.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_matern35_isotropic(): Derivatives of isotropic matern covariance function with smoothness 3.5
- d_matern45_isotropic(): Derivatives of isotropic matern covariance function with smoothness 3.5

Parameterization

The covariance parameter vector is (variance, range, nugget) = $(\sigma^2, \alpha, \tau^2)$, and the covariance function is parameterized as

$$M(x,y) = \sigma^2 (\sum_{j=0}^{3} c_j ||x-y||^j / \alpha^j) exp(-||x-y||/\alpha)$$

where c_0 = 1, c_1 = 1, c_2 = 2/5, c_3 = 1/15. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

---

matern35_scaledim — *Matern covariance function, smoothness = 3.5, different range parameter for each dimension*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, ..., range_d, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern35_scaledim(covparms, locs)

d_matern35_scaledim(covparms, locs)

d_matern45_scaledim(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range_1, ..., range_d, nugget)
locs	A matrix with n rows and d columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern35_scaledim()`: Derivatives with respect to parameters
- `d_matern45_scaledim()`: Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, ..., range_d, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2 (\sum_{j=0}^{3} c_j ||D^{-1}(x-y)||^j) exp(-||D^{-1}(x-y)||)$$

where c_0 = 1, c_1 = 1, c_2 = 2/5, c_3 = 1/15. where D is a diagonal matrix with (range_1, ..., range_d) on the diagonals. The nugget value $\sigma^2 \tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2 \tau^2$, not τ^2.

`matern45_isotropic` *Isotropic Matern covariance function, smoothness = 4.5*

Description

From a matrix of locations and covariance parameters of the form (variance, range, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern45_isotropic(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Parameterization

The covariance parameter vector is (variance, range, nugget) = $(\sigma^2, \alpha, \tau^2)$, and the covariance function is parameterized as

$$M(x,y) = \sigma^2(\sum_{j=0}^{4} c_j ||x-y||^j/\alpha^j)exp(-||x-y||/\alpha)$$

where c_0 = 1, c_1 = 1, c_2 = 3/7, c_3 = 2/21, c_4 = 1/105. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`matern45_scaledim` *Matern covariance function, smoothess = 3.5, different range parameter for each dimension*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, ..., range_d, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern45_scaledim(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, ..., range_d, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Parameterization

The covariance parameter vector is (variance, range_1, ..., range_d, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2(\sum_{j=0}^{4} c_j ||D^{-1}(x-y)||^j)exp(-||D^{-1}(x-y)||)$$

where c_0 = 1, c_1 = 1, c_2 = 3/7, c_3 = 2/21, c_4 = 1/105. where D is a diagonal matrix with (range_1, ..., range_d) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_anisotropic2D *Geometrically anisotropic Matern covariance function (two dimensions)*

Description

From a matrix of locations and covariance parameters of the form (variance, L11, L21, L22, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_anisotropic2D(covparms, locs)

d_matern_anisotropic2D(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, L11, L21, L22, smoothness, nugget)
locs	A matrix with n rows and 2 columns. Each row of locs is a point in R^2.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_matern_anisotropic2D(): Derivatives of anisotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, L11, L21, L22, smoothness, nugget) where L11, L21, L22, are the three non-zero entries of a lower-triangular matrix L. The covariances are

$$M(x,y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||Lx - Ly||)^\nu K_\nu(||Lx - Ly||)$$

This means that L11 is interpreted as an inverse range parameter in the first dimension. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`matern_anisotropic3D` *Geometrically anisotropic Matern covariance function (three dimensions)*

Description

From a matrix of locations and covariance parameters of the form (variance, L11, L21, L22, L31, L32, L33, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_anisotropic3D(covparms, locs)

d_matern_anisotropic3D(covparms, locs)

d_matern_anisotropic3D_alt(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, L11, L21, L22, L31, L32, L33, smoothness, nugget)
`locs`	A matrix with `n` rows and `3` columns. Each row of locs is a point in R^3.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern_anisotropic3D()`: Derivatives of anisotropic Matern covariance
- `d_matern_anisotropic3D_alt()`: Derivatives of anisotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, L11, L21, L22, L31, L32, L33, smoothness, nugget) where L11, L21, L22, L31, L32, L33 are the six non-zero entries of a lower-triangular matrix L. The covariances are

$$M(x,y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||Lx - Ly||)^\nu K_\nu(||Lx - Ly||)$$

This means that L11 is interpreted as an inverse range parameter in the first dimension. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_anisotropic3D_alt

Geometrically anisotropic Matern covariance function (three dimensions, alternate parameterization)

Description

From a matrix of locations and covariance parameters of the form (variance, B11, B12, B13, B22, B23, B33, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_anisotropic3D_alt(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, B11, B12, B13, B22, B23, B33, smoothness, nugget)
`locs`	A matrix with `n` rows and `3` columns. Each row of locs is a point in R^3.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Parameterization

The covariance parameter vector is (variance, B11, B12, B13, B22, B23, B33, smoothness, nugget) where B11, B12, B13, B22, B23, B33, transform the three coordinates as

$$u_1 = B11[x_1 + B12x_2 + (B13 + B12B23)x_3]$$

$$u_2 = B22[x_2 + B23x_3]$$

$$u_3 = B33[x_3]$$

NOTE: the u_1 transformation is different from previous versions of this function. NOTE: now (B13,B23) can be interpreted as a drift vector in space over time. Assuming x is transformed to u and y transformed to v, the covariances are

$$M(x, y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||u - v||)^\nu K_\nu(||u - v||)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`matern_categorical` *Isotropic Matern covariance function with random effects for categories*

Description

From a matrix of locations and covariance parameters of the form (variance, range, smoothness, category variance, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_categorical(covparms, locs)

d_matern_categorical(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, smoothness, category variance, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs gives a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern_categorical()`: Derivatives of isotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, range, smoothness, category variance, nugget) = $(\sigma^2, \alpha, \nu, c^2, \tau^2)$, and the covariance function is parameterized as

$$M(x, y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||x-y||/\alpha)^\nu K_\nu(||x-y||/\alpha)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. The category variance c^2 is added if two observation from same category NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_isotropic *Isotropic Matern covariance function*

Description

From a matrix of locations and covariance parameters of the form (variance, range, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_isotropic(covparms, locs)

d_matern_isotropic(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range, smoothness, nugget)
locs	A matrix with n rows and d columns. Each row of locs gives a point in R^d.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_matern_isotropic(): Derivatives of isotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, range, smoothness, nugget) = $(\sigma^2, \alpha, \nu, \tau^2)$, and the covariance function is parameterized as

$$M(x, y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||x - y||/\alpha)^\nu K_\nu(||x - y||/\alpha)$$

The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_nonstat_var *Isotropic Matern covariance function, nonstationary variances*

Description

From a matrix of locations and covariance parameters of the form (variance, range, smoothness, nugget, <nonstat variance parameters>), return the square matrix of all pairwise covariances.

Usage

```
matern_nonstat_var(covparms, Z)

d_matern_nonstat_var(covparms, Z)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range, smoothness, nugget, <nonstat variance parameters>). The number of nonstationary variance parameters should equal p.
Z	A matrix with n rows and 2 columns for spatial locations + p columns describing spatial basis functions. Each row of locs gives a point in R^2 (two dimensions only!) + the value of p spatial basis functions.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_matern_nonstat_var(): Derivatives with respect to parameters

Parameterization

This covariance function multiplies the isotropic Matern covariance by a nonstationary variance function. The form of the covariance is

$$C(x, y) = exp(\phi(x) + \phi(y))M(x, y)$$

where M(x,y) is the isotropic Matern covariance, and

$$\phi(x) = c_1\phi_1(x) + ... + c_p\phi_p(x)$$

where $\phi_1, ..., \phi_p$ are the spatial basis functions contained in the last p columns of Z, and $c_1, ..., c_p$ are the nonstationary variance parameters.

`matern_scaledim` *Matern covariance function, different range parameter for each dimension*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, ..., range_d, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_scaledim(covparms, locs)

d_matern_scaledim(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, ..., range_d, smoothness, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs is a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern_scaledim()`: Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, ..., range_d, smoothness, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||D^{-1}(x-y)||)^\nu K_\nu(||D^{-1}(x-y)||)$$

where D is a diagonal matrix with (range_1, ..., range_d) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_spacetime *Spatial-Temporal Matern covariance function*

Description

From a matrix of locations and covariance parameters of the form (variance, range_1, range_2, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_spacetime(covparms, locs)

d_matern_spacetime(covparms, locs)
```

Arguments

covparms	A vector with covariance parameters in the form (variance, range_1, range_2, smoothness, nugget). range_1 is the spatial range, and range_2 is the temporal range.
locs	A matrix with n rows and d+1 columns. Each row of locs is a point in R^(d+1). The first d columns should contain the spatial coordinates. The last column contains the times.

Value

A matrix with n rows and n columns, with the i,j entry containing the covariance between observations at locs[i,] and locs[j,].

Functions

- d_matern_spacetime(): Derivatives with respect to parameters

Parameterization

The covariance parameter vector is (variance, range_1, range_2, smoothness, nugget). The covariance function is parameterized as

$$M(x,y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(||D^{-1}(x-y)||)^\nu K_\nu(||D^{-1}(x-y)||)$$

where D is a diagonal matrix with (range_1, ..., range_1, range_2) on the diagonals. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_spacetime_categorical *Space-Time Matern covariance function with random effects for categories*

Description

From a matrix of locations and covariance parameters of the form (variance, spatial range, temporal range, smoothness, category, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_spacetime_categorical(covparms, locs)

d_matern_spacetime_categorical(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, spatial range, temporal range, smoothness, category, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs gives a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern_spacetime_categorical()`: Derivatives of isotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, range, smoothness, category, nugget) = $(\sigma^2, \alpha_1, \alpha_2, \nu, c^2, \tau^2)$, and the covariance function is parameterized as

$$d = (||x-y||^2/\alpha_1 + |s-t|^2/\alpha_2^2)^{1/2}$$

$$M(x,y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(d)^\nu K_\nu(d)$$

(x,s) and (y,t) are the space-time locations of a pair of observations. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. The category variance c^2 is added if two observation from same category NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

matern_spacetime_categorical_local
Space-Time Matern covariance function with local random effects for categories

Description

From a matrix of locations and covariance parameters of the form (variance, spatial range, temporal range, smoothness, cat variance, cat spatial range, cat temporal range, cat smoothness, nugget), return the square matrix of all pairwise covariances. This is the covariance for the following model for data from cateogory k

$$Y_k(x_i, t_i) = Z_0(x_i, t_i) + Z_k(x_i, t_i) + e_i$$

where Z_0 is Matern with parameters (variance,spatial range,temporal range,smoothness) and Z_1,...,Z_K are independent Materns with parameters (cat variance, cat spatial range, cat temporal range, cat smoothness), and e_1, ..., e_n are independent normals with variance (variance * nugget)

Usage

```
matern_spacetime_categorical_local(covparms, locs)

d_matern_spacetime_categorical_local(covparms, locs)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, spatial range, temporal range, smoothness, category, nugget)
`locs`	A matrix with `n` rows and `d` columns. Each row of locs gives a point in R^d.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `locs[i,]` and `locs[j,]`.

Functions

- `d_matern_spacetime_categorical_local()`: Derivatives of isotropic Matern covariance

Parameterization

The covariance parameter vector is (variance, range, smoothness, category, nugget) = $(\sigma^2, \alpha_1, \alpha_2, \nu, c^2, \tau^2)$, and the covariance function is parameterized as

$$d = (||x - y||^2/\alpha_1 + |s - t|^2/\alpha_2^2)^{1/2}$$

$$M(x, y) = \sigma^2 2^{1-\nu}/\Gamma(\nu)(d)^\nu K_\nu(d)$$

(x,s) and (y,t) are the space-time locations of a pair of observations. The nugget value $\sigma^2\tau^2$ is added to the diagonal of the covariance matrix. The category variance c^2 is added if two observation from same category NOTE: the nugget is $\sigma^2\tau^2$, not τ^2.

`matern_sphere` *Isotropic Matern covariance function on sphere*

Description

From a matrix of longitudes and latitudes and a vector covariance parameters of the form (variance, range, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_sphere(covparms, lonlat)

d_matern_sphere(covparms, lonlat)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, smoothness, nugget). Range parameter assumes that the sphere has radius 1 (units are radians).
`lonlat`	A matrix with `n` rows and one column with longitudes in (-180,180) and one column of latitudes in (-90,90). Each row of lonlat describes a point on the sphere.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlat[i,]` and `lonlat[j,]`.

Functions

- `d_matern_sphere()`: Derivatives with respect to parameters

Matern on Sphere Domain

The function first calculates the (x,y,z) 3D coordinates, and then inputs the resulting locations into `matern_isotropic`. This means that we construct covariances on the sphere by embedding the sphere in a 3D space. There has been some concern expressed in the literature that such embeddings may produce distortions. The source and nature of such distortions has never been articulated, and to date, no such distortions have been documented. Guinness and Fuentes (2016) argue that 3D embeddings produce reasonable models for data on spheres.

`matern_spheretime` *Matern covariance function on sphere x time*

Description

From a matrix of longitudes, latitudes, and times, and a vector covariance parameters of the form (variance, range_1, range_2, smoothness, nugget), return the square matrix of all pairwise covariances.

Usage

```
matern_spheretime(covparms, lonlattime)

d_matern_spheretime(covparms, lonlattime)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, range_2, smoothness, nugget), where range_1 is a spatial range (assuming sphere of radius 1), and range_2 is a temporal range.
`lonlattime`	A matrix with `n` rows and three columns: longitudes in (-180,180), latitudes in (-90,90), and times. Each row of lonlattime describes a point on the sphere x time.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlattime[i,]` and `lonlattime[j,]`.

Functions

- `d_matern_spheretime()`: Derivatives with respect to parameters

Covariances on spheres

The function first calculates the (x,y,z) 3D coordinates, and then inputs the resulting locations into `matern_spacetime`. This means that we construct covariances on the sphere by embedding the sphere in a 3D space. There has been some concern expressed in the literature that such embeddings may produce distortions. The source and nature of such distortions has never been articulated, and to date, no such distortions have been documented. Guinness and Fuentes (2016) argue that 3D embeddings produce reasonable models for data on spheres.

`matern_spheretime_warp` *Deformed Matern covariance function on sphere*

Description

From a matrix of longitudes, latitudes, times, and a vector covariance parameters of the form (variance, range_1, range_2, smoothness, nugget, <5 warping parameters>), return the square matrix of all pairwise covariances.

Usage

```
matern_spheretime_warp(covparms, lonlattime)

d_matern_spheretime_warp(covparms, lonlattime)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range_1, range_2, smoothness, nugget, <5 warping parameters>). range_1 is a spatial range parameter that assumes that the sphere has radius 1 (units are radians). range_2 is a temporal range parameter.
`lonlattime`	A matrix with `n` rows and three columns: longitudes in (-180,180), latitudes in (-90,90), and times. Each row of lonlattime describes a point on the sphere x time.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlat[i,]` and `lonlat[j,]`.

Functions

- `d_matern_spheretime_warp()`: Derivatives with respect to parameters

Warpings

The function first calculates the (x,y,z) 3D coordinates, and then "warps" the locations to $(x, y, z) + \Phi(x, y, z)$, where Φ is a warping function composed of gradients of spherical harmonic functions of degree 2. See Guinness (2019, "Gaussian Process Learning via Fisher Scoring of Vecchia's Approximation") for details. The warped locations are input into `matern_spacetime`. The function does not do temporal warping.

matern_sphere_warp *Deformed Matern covariance function on sphere*

Description

From a matrix of longitudes and latitudes and a vector covariance parameters of the form (variance, range, smoothness, nugget, <5 warping parameters>), return the square matrix of all pairwise covariances.

Usage

```
matern_sphere_warp(covparms, lonlat)

d_matern_sphere_warp(covparms, lonlat)
```

Arguments

`covparms`	A vector with covariance parameters in the form (variance, range, smoothness, nugget, <5 warping parameters>). Range parameter assumes that the sphere has radius 1 (units are radians).
`lonlat`	A matrix with `n` rows and one column with longitudes in (-180,180) and one column of latitudes in (-90,90). Each row of lonlat describes a point on the sphere.

Value

A matrix with `n` rows and `n` columns, with the i,j entry containing the covariance between observations at `lonlat[i,]` and `lonlat[j,]`.

Functions

- `d_matern_sphere_warp()`: Derivatives with respect to parameters.

Warpings

The function first calculates the (x,y,z) 3D coordinates, and then "warps" the locations to $(x, y, z) + \Phi(x, y, z)$, where Φ is a warping function composed of gradients of spherical harmonic functions of degree 2. See Guinness (2019, "Gaussian Process Learning via Fisher Scoring of Vecchia's Approximation") for details. The warped locations are input into `matern_isotropic`.

order_coordinate *Sorted coordinate ordering*

Description

Return the ordering of locations sorted along one of the coordinates or the sum of multiple coordinates

Usage

```
order_coordinate(locs, coordinate)
```

Arguments

locs	A matrix of locations. Each row of locs contains a location, which can be a point in Euclidean space R^d, a point in space-time R^d x T, a longitude and latitude (in degrees) giving a point on the sphere, or a longitude, latitude, and time giving a point in the sphere-time domain.
coordinate	integer or vector of integers in (1,...,d). If a single integer, coordinates are ordered along that coordinate. If multiple integers, coordinates are ordered according to the sum of specified coordinate values. For example, when d=2, coordinate = c(1,2) orders from bottom left to top right.

Value

A vector of indices giving the ordering, i.e. the first element of this vector is the index of the first location.

Examples

```
n <- 100             # Number of locations
d <- 2               # dimension of domain
locs <- matrix( runif(n*d), n, d )
ord1 <- order_coordinate(locs, 1 )
ord12 <- order_coordinate(locs, c(1,2) )
```

order_dist_to_point *Distance to specified point ordering*

Description

Return the ordering of locations increasing in their distance to some specified location

Usage

```
order_dist_to_point(locs, loc0, lonlat = FALSE)
```

Arguments

- `locs` A matrix of locations. Each row of `locs` contains a location, which can be a point in Euclidean space R^d, a point in space-time R^d x T, a longitude and latitude (in degrees) giving a point on the sphere, or a longitude, latitude, and time giving a point in the sphere-time domain.
- `loc0` A vector containing a single location in R^d.
- `lonlat` TRUE/FALSE whether locations are longitudes and latitudes.

Value

A vector of indices giving the ordering, i.e. the first element of this vector is the index of the location nearest to `loc0`.

Examples

```
n <- 100             # Number of locations
d <- 2               # dimension of domain
locs <- matrix( runif(n*d), n, d )
loc0 <- c(1/2,1/2)
ord <- order_dist_to_point(locs,loc0)
```

`order_maxmin` *Maximum minimum distance ordering*

Description

Return the indices of an approximation to the maximum minimum distance ordering. A point in the center is chosen first, and then each successive point is chosen to maximize the minimum distance to previously selected points

Usage

```
order_maxmin(locs, lonlat = FALSE, space_time = FALSE, st_scale = NULL)
```

Arguments

- `locs` A matrix of locations. Each row of `locs` contains a location, which can be a point in Euclidean space R^d, a point in space-time R^d x T, a longitude and latitude (in degrees) giving a point on the sphere, or a longitude, latitude, and time giving a point in the sphere-time domain.
- `lonlat` TRUE/FALSE whether locations are longitudes and latitudes.
- `space_time` TRUE if locations are euclidean space-time locations, FALSE otherwise. If set to TRUE, temporal dimension is ignored.
- `st_scale` two-vector giving the amount by which the spatial and temporal coordinates are scaled. If `NULL`, the function uses the locations to automatically select a scaling. If set to FALSE, temporal dimension treated as another spatial dimension (not recommended).

Value

A vector of indices giving the ordering, i.e. the first element of this vector is the index of the first location.

Examples

```
# planar coordinates
nvec <- c(50,50)
locs <- as.matrix( expand.grid( 1:nvec[1]/nvec[1], 1:nvec[2]/nvec[2] ) )
ord <- order_maxmin(locs)
par(mfrow=c(1,3))
plot( locs[ord[1:100],1], locs[ord[1:100],2], xlim = c(0,1), ylim = c(0,1) )
plot( locs[ord[1:300],1], locs[ord[1:300],2], xlim = c(0,1), ylim = c(0,1) )
plot( locs[ord[1:900],1], locs[ord[1:900],2], xlim = c(0,1), ylim = c(0,1) )

# longitude/latitude coordinates (sphere)
latvals <- seq(-80, 80, length.out = 40 )
lonvals <- seq( 0, 360, length.out = 81 )[1:80]
locs <- as.matrix( expand.grid( lonvals, latvals ) )
ord <- order_maxmin(locs, lonlat = TRUE)
par(mfrow=c(1,3))
plot( locs[ord[1:100],1], locs[ord[1:100],2], xlim = c(0,360), ylim = c(-90,90) )
plot( locs[ord[1:300],1], locs[ord[1:300],2], xlim = c(0,360), ylim = c(-90,90) )
plot( locs[ord[1:900],1], locs[ord[1:900],2], xlim = c(0,360), ylim = c(-90,90) )
```

---

order_middleout *Middle-out ordering*

Description

Return the ordering of locations increasing in their distance to the average location

Usage

```
order_middleout(locs, lonlat = FALSE)
```

Arguments

locs	A matrix of locations. Each row of locs contains a location, which can be a point in Euclidean space R^d, a point in space-time R^d x T, a longitude and latitude (in degrees) giving a point on the sphere, or a longitude, latitude, and time giving a point in the sphere-time domain.
lonlat	TRUE/FALSE whether locations are longitudes and latitudes.

Value

A vector of indices giving the ordering, i.e. the first element of this vector is the index of the location nearest the center.

Examples

```
n <- 100             # Number of locations
d <- 2               # dimension of domain
locs <- matrix( runif(n*d), n, d )
ord <- order_middleout(locs)
```

---

pen_hi *penalize large values of parameter: penalty, 1st deriative, 2nd derivative*

Description

penalize large values of parameter: penalty, 1st deriative, 2nd derivative

Usage

```
pen_hi(x, tt, aa)

dpen_hi(x, tt, aa)

ddpen_hi(x, tt, aa)
```

Arguments

x	argument to penalty
tt	scale parameter of penalty
aa	location parameter of penalty

---

pen_lo *penalize small values of parameter: penalty, 1st deriative, 2nd derivative*

Description

penalize small values of parameter: penalty, 1st deriative, 2nd derivative

Usage

```
pen_lo(x, tt, aa)

dpen_lo(x, tt, aa)

ddpen_lo(x, tt, aa)
```

Arguments

x	argument to penalty
tt	scale parameter of penalty
aa	location parameter of penalty

---

pen_loglo *penalize small values of log parameter: penalty, 1st deriative, 2nd derivative*

Description

penalize small values of log parameter: penalty, 1st deriative, 2nd derivative

Usage

```
pen_loglo(x, tt, aa)

dpen_loglo(x, tt, aa)

ddpen_loglo(x, tt, aa)
```

Arguments

x	argument to penalty
tt	scale parameter of penalty
aa	location parameter of penalty

---

predictions *Compute Gaussian process predictions using Vecchia's approximations*

Description

With the prediction locations ordered after the observation locations, an approximation for the inverse Cholesky of the covariance matrix is computed, and standard formulas are applied to obtain the conditional expectation.

Usage

```
predictions(
  fit = NULL,
  locs_pred,
  X_pred,
  y_obs = fit$y,
  locs_obs = fit$locs,
  X_obs = fit$X,
  beta = fit$betahat,
  covparms = fit$covparms,
  covfun_name = fit$covfun_name,
  m = 60,
  reorder = TRUE,
  st_scale = NULL
)
```

Arguments

`fit`	GpGp_fit object, the result of `fit_model`
`locs_pred`	prediction locations
`X_pred`	Design matrix for predictions
`y_obs`	Observations associated with locs_obs
`locs_obs`	observation locations
`X_obs`	Design matrix for observations
`beta`	Linear mean parameters
`covparms`	Covariance parameters
`covfun_name`	Name of covariance function
`m`	Number of nearest neighbors to use
`reorder`	TRUE/FALSE for whether reordering should be done. This should generally be kept at TRUE, unless testing out the effect of reordering.
`st_scale`	amount by which to scale the spatial and temporal dimensions for the purpose of selecting neighbors. We recommend setting this manually when using a spatial-temporal covariance function. When `lonlat = TRUE`, spatial scale is in radians (earth radius = 1).

Details

We can specify either a GpGp_fit object (the result of `fit_model`), OR manually enter the covariance function and parameters, the observations, observation locations, and design matrix. We must specify the prediction locations and the prediction design matrix.

sph_grad_xyz — *compute gradient of spherical harmonics functions*

Description

compute gradient of spherical harmonics functions

Usage

```
sph_grad_xyz(xyz, Lmax)
```

Arguments

`xyz`	xyz coordinates of locations on sphere
`Lmax`	largest degree of spherical harmonics. Current only Lmax=2 supported

summary.GpGp_fit — *Print summary of GpGp fit*

Description

Print summary of GpGp fit

Usage

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

Arguments

`object`	Object of class "GpGp_fit", usually the return value from `fit_model`
`...`	additional arguments, for compatability with S3 generic 'summary'

test_likelihood_object
test likelihood object for NA or Inf values

Description

test likelihood object for NA or Inf values

Usage

```
test_likelihood_object(likobj)
```

Arguments

likobj	likelihood object

vecchia_grouped_meanzero_loglik
Grouped Vecchia approximation to the Gaussian loglikelihood, zero mean

Description

This function returns a grouped version (Guinness, 2018) of Vecchia's (1988) approximation to the Gaussian loglikelihood. The approximation modifies the ordered conditional specification of the joint density; rather than each term in the product conditioning on all previous observations, each term conditions on a small subset of previous observations.

Usage

```
vecchia_grouped_meanzero_loglik(covparms, covfun_name, y, locs, NNlist)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
y	vector of response values
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNlist	A neighbor list object, the output from group_obs.

Value

a list containing

- loglik: the loglikelihood

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
covparms <- c(2, 0.2, 0.75, 0)
y <- fast_Gp_sim(covparms, "matern_isotropic", locs, 50 )
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
NNlist <- group_obs(NNarray)
#loglik <- vecchia_grouped_meanzero_loglik( covparms, "matern_isotropic", y, locs, NNlist )
```

---

vecchia_grouped_profbeta_loglik

Grouped Vecchia approximation, profiled regression coefficients

---

Description

This function returns a grouped version (Guinness, 2018) of Vecchia's (1988) approximation to the Gaussian loglikelihood and the profile likelihood estimate of the regression coefficients. The approximation modifies the ordered conditional specification of the joint density; rather than each term in the product conditioning on all previous observations, each term conditions on a small subset of previous observations.

Usage

```
vecchia_grouped_profbeta_loglik(covparms, covfun_name, y, X, locs, NNlist)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
y	vector of response values
X	Design matrix of covariates. Row i of X contains the covariates for the observation at row i of locs.
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNlist	A neighbor list object, the output from group_obs.

Value

a list containing

- loglik: the loglikelihood
- betahat: profile likelihood estimate of regression coefficients
- betainfo: information matrix for betahat.

The covariance matrix for $betahat is the inverse of $betainfo.

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
X <- cbind(rep(1,n),locs[,2])
covparms <- c(2, 0.2, 0.75, 0)
y <- fast_Gp_sim(covparms, "matern_isotropic", locs, 50 )
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
NNlist <- group_obs(NNarray)
#loglik <- vecchia_grouped_profbeta_loglik(
#    covparms, "matern_isotropic", y, X, locs, NNlist )
```

---

vecchia_grouped_profbeta_loglik_grad_info

Grouped Vecchia loglikelihood, gradient, Fisher information

Description

This function returns a grouped version (Guinness, 2018) of Vecchia's (1988) approximation to the Gaussian loglikelihood, the gradient, and Fisher information, and the profile likelihood estimate of the regression coefficients. The approximation modifies the ordered conditional specification of the joint density; rather than each term in the product conditioning on all previous observations, each term conditions on a small subset of previous observations.

Usage

```
vecchia_grouped_profbeta_loglik_grad_info(
  covparms,
  covfun_name,
  y,
  X,
  locs,
  NNlist
)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
y	vector of response values
X	Design matrix of covariates. Row i of X contains the covariates for the observation at row i of locs.
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNlist	A neighbor list object, the output from group_obs.

Value

a list containing

- loglik: the loglikelihood
- grad: gradient with respect to covariance parameters
- info: Fisher information for covariance parameters
- betahat: profile likelihood estimate of regression coefs
- betainfo: information matrix for betahat.

The covariance matrix for $betahat is the inverse of $betainfo.

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
X <- cbind(rep(1,n),locs[,2])
covparms <- c(2, 0.2, 0.75, 0)
y <- fast_Gp_sim(covparms, "matern_isotropic", locs, 50 )
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
NNlist <- group_obs(NNarray)
#loglik <- vecchia_grouped_profbeta_loglik_grad_info(
#    covparms, "matern_isotropic", y, X, locs, NNlist )
```

---

vecchia_Linv — *Entries of inverse Cholesky approximation*

Description

This function returns the entries of the inverse Cholesky factor of the covariance matrix implied by Vecchia's approximation. For return matrix Linv, Linv[i,] contains the non-zero entries of row i of the inverse Cholesky matrix. The columns of the non-zero entries are specified in NNarray[i,].

Usage

```
vecchia_Linv(covparms, covfun_name, locs, NNarray, start_ind = 1L)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.
start_ind	Compute entries of Linv only for rows start_ind until the last row.

Value

matrix containing entries of inverse Cholesky

Examples

```
n1 <- 40
n2 <- 40
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
covparms <- c(2, 0.2, 0.75, 0)
NNarray <- find_ordered_nn(locs,20)
Linv <- vecchia_Linv(covparms, "matern_isotropic", locs, NNarray)
```

---

vecchia_meanzero_loglik

Vecchia's approximation to the Gaussian loglikelihood, zero mean

Description

This function returns Vecchia's (1988) approximation to the Gaussian loglikelihood. The approximation modifies the ordered conditional specification of the joint density; rather than each term in the product conditioning on all previous observations, each term conditions on a small subset of previous observations.

Usage

```
vecchia_meanzero_loglik(covparms, covfun_name, y, locs, NNarray)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
y	vector of response values
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

a list containing

- loglik: the loglikelihood

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
covparms <- c(2, 0.2, 0.75, 0)
y <- fast_Gp_sim(covparms, "matern_isotropic", locs, 50 )
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
#loglik <- vecchia_meanzero_loglik( covparms, "matern_isotropic", y, locs, NNarray )
```

---

vecchia_profbeta_loglik

Vecchia's approximation to the Gaussian loglikelihood, with profiled regression coefficients.

Description

This function returns Vecchia's (1988) approximation to the Gaussian loglikelihood, profiling out the regression coefficients. The approximation modifies the ordered conditional specification of the joint density; rather than each term in the product conditioning on all previous observations, each term conditions on a small subset of previous observations.

Usage

```
vecchia_profbeta_loglik(covparms, covfun_name, y, X, locs, NNarray)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
y	vector of response values
X	Design matrix of covariates. Row i of X contains the covariates for the observation at row i of locs.
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

a list containing

- loglik: the loglikelihood
- betahat: profile likelihood estimate of regression coefficients
- betainfo: information matrix for betahat.

The covariance matrix for $betahat is the inverse of $betainfo.

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
X <- cbind(rep(1,n),locs[,2])
covparms <- c(2, 0.2, 0.75, 0)
y <- X %*% c(1,2) + fast_Gp_sim(covparms, "matern_isotropic", locs, 50 )
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
#loglik <- vecchia_profbeta_loglik( covparms, "matern_isotropic", y, X, locs, NNarray )
```

---

vecchia_profbeta_loglik_grad_info

Vecchia's loglikelihood, gradient, and Fisher information

Description

This function returns Vecchia's (1988) approximation to the Gaussian loglikelihood, profiling out the regression coefficients, and returning the gradient and Fisher information. Vecchia's approximation modifies the ordered conditional specification of the joint density; rather than each term in the product conditioning on all previous observations, each term conditions on a small subset of previous observations.

Usage

```
vecchia_profbeta_loglik_grad_info(covparms, covfun_name, y, X, locs, NNarray)
```

Arguments

covparms	A vector of covariance parameters appropriate for the specified covariance function
covfun_name	See GpGp for information about covariance functions.
y	vector of response values
X	Design matrix of covariates. Row i of X contains the covariates for the observation at row i of locs.
locs	matrix of locations. Row i of locs specifies the location of element i of y, and so the length of y should equal the number of rows of locs.
NNarray	A matrix of indices, usually the output from find_ordered_nn. Row i contains the indices of the observations that observation i conditions on. By convention, the first element of row i is i.

Value

A list containing

- loglik: the loglikelihood
- grad: gradient with respect to covariance parameters
- info: Fisher information for covariance parameters
- betahat: profile likelihood estimate of regression coefs
- betainfo: information matrix for betahat.

The covariance matrix for $betahat is the inverse of $betainfo.

Examples

```
n1 <- 20
n2 <- 20
n <- n1*n2
locs <- as.matrix( expand.grid( (1:n1)/n1, (1:n2)/n2 ) )
X <- cbind(rep(1,n),locs[,2])
covparms <- c(2, 0.2, 0.75, 0)
y <- X %*% c(1,2) + fast_Gp_sim(covparms, "matern_isotropic", locs, 50 )
ord <- order_maxmin(locs)
NNarray <- find_ordered_nn(locs,20)
#loglik <- vecchia_profbeta_loglik_grad_info( covparms, "matern_isotropic",
#    y, X, locs, NNarray )
```

Index