# Package 'BayesNSGP'

April 5, 2019

**Title** Bayesian Analysis of Non-Stationary Gaussian Process Models

**Description** MARK: UP FOR FILLING THIS IN? ONE SHORT PARAGRAPH.

**Version** 0.1.0

**Date** 2019-01-20

**Maintainer** Daniel Turek <dbt1@williams.edu>

**Author** Mark Risser, Daniel Turek

**Depends** R (>= 3.4.0)

**Imports** FNN,Matrix,methods,nimble,StatMatch

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.0

## R topics documented:

## `calcQF` *Calculate the Gaussian quadratic form for the NNGP approximation*

### Description

`calcQF` calculates the quadratic form in the multivariate Gaussian based on the NNGP approximation, for a specific parameter combination. The quadratic form is `t(u)C^{-1}v`.

### Usage

```
calcQF(u, v, AD, nID)
```

### Arguments

| | |
|---|---|
| `u` | Vector; left product. |
| `v` | Vector; right product |
| `AD` | N x (k+1) matrix; the first k columns are the 'A' matrix, and the last column is the 'D' vector. Represents the Cholesky of `C^{-1}`. |
| `nID` | N x k matrix of neighbor indices. |

### Value

A list with two components: (1) an N x 2 array containing the same spatial coordinates, ordered by MMD, and (2) the same thing, but with any NA values removed.

### Examples

```
# TODO
```

## `calculateAD_ns` *Calculate A and D matrices for the NNGP approximation*

### Description

`calculateAD_ns` calculates A and D matrices (the Cholesky of the precision matrix) needed for the NNGP approximation.

### Usage

```
calculateAD_ns(dist1_3d, dist2_3d, dist12_3d, Sigma11, Sigma22, Sigma12,
  log_sigma_vec, log_tau_vec, nID, N, k, nu, d)
```

## Arguments

| | |
|---|---|
| `dist1_3d` | N x (k+1) x (k+1) array of distances in the x-coordinate direction. |
| `dist2_3d` | N x (k+1) x (k+1) array of distances in the y-coordinate direction. |
| `dist12_3d` | N x (k+1) x (k+1) array of cross-distances. |
| `Sigma11` | N-vector; 1-1 element of the Sigma() process. |
| `Sigma22` | N-vector; 2-2 element of the Sigma() process. |
| `Sigma12` | N-vector; 1-2 element of the Sigma() process. |
| `log_sigma_vec` | N-vector; process standard deviation values. |
| `log_tau_vec` | N-vector; nugget standard deviation values. |
| `nID` | N x k matrix of neighbor indices. |
| `N` | Scalar; number of data measurements. |
| `k` | Scalar; number of nearest neighbors. |
| `nu` | Scalar; Matern smoothness parameter. |
| `d` | TODO |

## Value

A N x (k+1) matrix; the first k columns are the 'A' matrix, and the last column is the 'D' vector.

## Examples

```
# TODO
```

---

`calculateU_ns` *Calculate the (sparse) matrix U*

---

## Description

`calculateU_ns` calculates the (sparse) matrix U (i.e., the Cholesky of the inverse covariance matrix) using a nonstationary covariance function. The output only contains non-zero values and is stored as three vectors: (1) the row indices, (2) the column indices, and (3) the non-zero values. NOTE: this code assumes the all inputs correspond to the ORDERED locations.

## Usage

```
calculateU_ns(dist1_3d, dist2_3d, dist12_3d, Sigma11, Sigma22, Sigma12,
  log_sigma_vec, log_tau_vec, nu, nID, cond_on_y, N, k, d, M = 0)
```

## Arguments

| | |
|---|---|
| `dist1_3d` | N x (k+1) x (k+1) array of distances in the x-coordinate direction. |
| `dist2_3d` | N x (k+1) x (k+1) array of distances in the y-coordinate direction. |
| `dist12_3d` | N x (k+1) x (k+1) array of cross-distances. |
| `Sigma11` | N-vector; 1-1 element of the Sigma() process. |
| `Sigma22` | N-vector; 2-2 element of the Sigma() process. |

| | |
|---|---|
| `Sigma12` | N-vector; 1-2 element of the Sigma() process. |
| `log_sigma_vec` | N-vector; process standard deviation values. |
| `log_tau_vec` | N-vector; nugget standard deviation values. |
| `nu` | Scalar; Matern smoothness parameter. |
| `nID` | N x k matrix of (ordered) neighbor indices. |
| `cond_on_y` | A matrix indicating whether the conditioning set for each (ordered) location is on the latent process (y, `1`) or the observed values (z, `0`). Calculated in `sgvSetup`. |
| `N` | Scalar; number of data measurements. |
| `k` | Scalar; number of nearest neighbors. |
| `d` | TODO |
| `M` | TODO |

### Value

TODO

### Examples

```
# TODO
```

---

## `conditionLatentObs` *Assign conditioning sets for the SGV approximation*

---

### Description

`conditionLatentObs` assigns q_y(i) vs q_z(i) following Section 5.1 in Katzfuss and Guinness (2018). This function only needs to be run once per SGV analysis.

### Usage

```
conditionLatentObs(nID, locs_ord, N)
```

### Arguments

| | |
|---|---|
| `nID` | N x k matrix of neighbor indices. |
| `locs_ord` | N x 2 matrix of locations. |
| `N` | Scalar; number of locations (observed only!). |

### Value

A matrix indicating whether the conditioning set for each location is on the latent process (y, `1`) or the observed values (z, `0`).

### Examples

```
# TODO
```

| | |
|---|---|
| `determineNeighbors` | *Determine the k-nearest neighbors for each spatial coordinate.* |

## Description

`determineNeighbors` returns an N x k matrix of the nearest neighbors for spatial locations s, with the ith row giving indices of the k nearest neighbors to the ith location, which are selected from among the 1,...(i-1) other spatial locations. The first row is -1's, since the first location has no neighbors. The i=2 through i=(k+1) rows each necessarily contain 1:i.

## Usage

```
determineNeighbors(s, k)
```

## Arguments

| | |
|---|---|
| `s` | N x 2 array of N 2-dimensional (x,y) spatial coordinates. |
| `k` | Scalar; number of neighbors |

## Value

An N x k matrix of nearest neighbor indices

## Examples

```
# TODO
```

| | |
|---|---|
| `dmnorm_nngp` | *Function for the evaluating the NNGP approximate density.* |

## Description

`dmnorm_nngp` (and `rmnorm_nngp`) calculate the approximate NNGP likelihood for a fixed set of parameters (i.e., A and D matrices). Finally, the distributions must be registered within `nimble`.

## Usage

```
dmnorm_nngp(x, mean, AD, nID, N, k, log)
```

## Arguments

| | |
|---|---|
| `x` | N-vector of data. |
| `mean` | N-vector with current values of the mean |
| `AD` | N x (k+1) matrix; the first k columns are the 'A' matrix, and the last column is the 'D' vector. |
| `nID` | N x k matrix of neighbor indices. |
| `N` | Scalar; number of data measurements. |
| `k` | Scalar; number of nearest neighbors. |
| `log` | Scalar; should the density be on the log scale (1) or not (0). |

## Value

The NNGP approximate density.

## Examples

```
# TODO
```

---

`dmnorm_sgv` *Function for the evaluating the SGV approximate density.*

---

## Description

`dmnorm_sgv` (and `rmnorm_sgv`) calculate the approximate SGV likelihood for a fixed set of parameters (i.e., the U matrix). Finally, the distributions must be registered within `nimble`.

## Usage

```
dmnorm_sgv(x, mean, U, N, k, log = 1)
```

## Arguments

| | |
|---|---|
| `x` | TODO |
| `mean` | TODO |
| `U` | TODO |
| `N` | TODO |
| `k` | TODO |
| `log` | TODO |

## Value

TODO

## Examples

```
# TODO
```

---

`inverseEigen` *Calculate covariance elements based on eigendecomposition components*

---

## Description

`inverseEigen` calculates the inverse eigendecomposition – in other words, the covariance elements based on the eigenvalues and vectors (see Paciorek and Schervish, 2006, for details on the parameterization). The function is coded as a `nimbleFunction` (see the `nimble` package) but can also be used as a regular R function.

## Usage

```
inverseEigen(eigen_comp1, eigen_comp2, eigen_comp3, which_Sigma)
```

## Arguments

| | |
|---|---|
| `eigen_comp1` | N-vector; contains values of the log of the second anisotropy eigenvalue for a set of locations. |
| `eigen_comp2` | N-vector; contains values of the first eigenvector component for a set of locations. |
| `eigen_comp3` | N-vector; contains values of the second eigenvector component for a set of locations. |
| `which_Sigma` | Scalar; one of `(1,2,3)`, corresponding to which covariance component should be calculated (Sigma11, Sigma22, or Sigma12, respectively). |

## Value

A correlation matrix for a fixed set of stations and fixed parameter values.

## Examples

```
# TODO
```

---

`matern_corr` *Calculate a stationary Matern correlation matrix*

---

## Description

`matern_corr` calculates a stationary Matern correlation matrix for a fixed set of locations, based on a range and smoothness parameter. This function is primarily used for the "npGP" and "approxGP" models. The function is coded as a `nimbleFunction` (see the `nimble` package) but can also be used as a regular R function.

## Usage

```
matern_corr(dist, rho, nu)
```

### Arguments

| | |
|---|---|
| `dist` | N x N matrix; contains values of pairwise Euclidean distances in the x-y plane. |
| `rho` | Scalar; "range" parameter used to rescale distances |
| `nu` | Scalar; Matern smoothness parameter. `nu = 0.5` corresponds to the Exponential correlation; `nu = Inf` corresponds to the Gaussian correlation function. |

### Value

A correlation matrix for a fixed set of stations and fixed parameter values.

### Examples

```
# TODO
```

## `nsCorr` *Calculate a nonstationary Matern correlation matrix*

### Description

`nsCorr` calculates a nonstationary correlation matrix for a fixed set of locations, based on vectors of the unique anisotropy parameters for each station. Since the correlation function uses a spatially-varying Mahalanobis distance, this function requires coordinate- specific distance matrices (see below). The function is coded as a `nimbleFunction` (see the `nimble` package) but can also be used as a regular R function.

### Usage

```
nsCorr(dist1_sq, dist2_sq, dist12, Sigma11, Sigma22, Sigma12, nu, d)
```

### Arguments

| | |
|---|---|
| `dist1_sq` | N x N matrix; contains values of pairwise squared distances in the x-coordinate. |
| `dist2_sq` | N x N matrix; contains values of pairwise squared distances in the y-coordinate. |
| `dist12` | N x N matrix; contains values of pairwise signed cross- distances between the x- and y-coordinates. The sign of each element is important; see `nsDist` function for the details of this calculation. in the x-coordinate. |
| `Sigma11` | Vector of length N; contains the 1-1 element of the anisotropy process for each station. |
| `Sigma22` | Vector of length N; contains the 2-2 element of the anisotropy process for each station. |
| `Sigma12` | Vector of length N; contains the 1-2 element of the anisotropy process for each station. |
| `nu` | Scalar; Matern smoothness parameter. `nu = 0.5` corresponds to the Exponential correlation; `nu = Inf` corresponds to the Gaussian correlation function. |
| `d` | TODO |

## Value

A correlation matrix for a fixed set of stations and fixed parameter values.

## Examples

```
# TODO
```

---

`nsCrosscorr` *Calculate a nonstationary Matern cross-correlation matrix*

---

## Description

`nsCrosscorr` calculates a nonstationary cross-correlation matrix between two fixed sets of locations (a prediction set with M locations, and the observed set with N locations), based on vectors of the unique anisotropy parameters for each station. Since the correlation function uses a spatially-varying Mahalanobis distance, this function requires coordinate- specific distance matrices (see below). The function is coded as a `nimbleFunction` (see the `nimble` package) but can also be used as a regular R function.

## Usage

```
nsCrosscorr(Xdist1_sq, Xdist2_sq, Xdist12, Sigma11, Sigma22, Sigma12,
  PSigma11, PSigma22, PSigma12, nu, d)
```

## Arguments

| | |
|---|---|
| `Xdist1_sq` | M x N matrix; contains values of pairwise squared cross-distances in the x-coordinate. |
| `Xdist2_sq` | M x N matrix; contains values of pairwise squared cross-distances in the y-coordinate. |
| `Xdist12` | M x N matrix; contains values of pairwise signed cross/cross- distances between the x- and y-coordinates. The sign of each element is important; see `nsDist` function for the details of this calculation. in the x-coordinate. |
| `Sigma11` | Vector of length N; contains the 1-1 element of the anisotropy process for each observed location. |
| `Sigma22` | Vector of length N; contains the 2-2 element of the anisotropy process for each observed location. |
| `Sigma12` | Vector of length N; contains the 1-2 element of the anisotropy process for each observed location. |
| `PSigma11` | Vector of length N; contains the 1-1 element of the anisotropy process for each prediction location. |
| `PSigma22` | Vector of length N; contains the 2-2 element of the anisotropy process for each prediction location. |
| `PSigma12` | Vector of length N; contains the 1-2 element of the anisotropy process for each prediction location. |
| `nu` | Scalar; Matern smoothness parameter. `nu = 0.5` corresponds to the Exponential correlation; `nu = Inf` corresponds to the Gaussian correlation function. |
| `d` | TODO |

## Value

A cross-correlation matrix for two fixed sets of stations and fixed parameter values.

## Examples

```
# TODO
```

---

`nsCrossdist` *Calculate coordinate-specific cross-distance matrices*

---

## Description

`nsCrossdist` calculates coordinate-specific cross distances in x, y, and x-y for use in the nonstationary cross-correlation calculation. This function is useful for calculating posterior predictions.

## Usage

```
nsCrossdist(coords, Pcoords, scale_factor = NULL, isotropic = FALSE)
```

## Arguments

| | |
|---|---|
| `coords` | N x 2 matrix; contains x-y coordinates of station (observed) locations. |
| `Pcoords` | M x 2 matrix; contains x-y coordinates of prediction locations. |
| `scale_factor` | Scalar; optional argument for re-scaling the distances. |
| `isotropic` | TODO |

## Value

A list of distances matrices, with the following components:

| | |
|---|---|
| `dist1_sq` | M x N matrix; contains values of pairwise squared cross- distances in the x-coordinate. |
| `dist2_sq` | M x N matrix; contains values of pairwise squared cross- distances in the y-coordinate. |
| `dist12` | M x N matrix; contains values of pairwise signed cross- distances between the x- and y-coordinates. |
| `scale_factor` | Value of the scale factor used to rescale distances. |

## Examples

```
# TODO
```

## nsCrossdist3d — *Calculate coordinate-specific distance matrices, only for nearest neighbors and store in an array*

### Description

nsCrossdist3d generates and returns new 3-dimensional arrays containing the former dist1_sq, dist2_s1, and dist12 matrices, but only as needed for the k nearest-neighbors of each location. these 3D matrices (dist1_3d, dist2_3d, and dist12_3d) are used in the new implementation of calculateAD_ns().

### Usage

```
nsCrossdist3d(coords, predCoords, P_nID, scale_factor = NULL,
  isotropic = FALSE)
```

### Arguments

| | |
|---|---|
| coords | N x 2 matrix; contains the x-y coordinates of stations. |
| predCoords | TODO |
| P_nID | N x k matrix; contains indices of nearest neighbors. |
| scale_factor | Scalar; optional argument for re-scaling the distances. |
| isotropic | Logical; indicates whether distances should be calculated separately for each coordinate dimension (FALSE) or simultaneously for all coordinate dimensions (TRUE). `isotropic = TRUE` can only be used for two-dimensional coordinate systems. |

### Value

Arrays with nearest neighbor distances in each coordinate direction.

### Examples

```
# TODO
```

## nsDist — *Calculate coordinate-specific distance matrices*

### Description

nsDist calculates x, y, and x-y distances for use in the nonstationary correlation calculation. The sign of the cross-distance is important. The function contains an optional argument for re-scaling the distances such that the coordinates lie in a square.

### Usage

```
nsDist(coords, scale_factor = NULL, isotropic = FALSE)
```

## Arguments

| | |
|---|---|
| `coords` | N x 2 matrix; contains the x-y coordinates of stations |
| `scale_factor` | Scalar; optional argument for re-scaling the distances. |
| `isotropic` | Logical; indicates whether distances should be calculated separately for each coordinate dimension (FALSE) or simultaneously for all coordinate dimensions (TRUE). `isotropic = TRUE` can only be used for two-dimensional coordinate systems. |

## Value

A list of distances matrices, with the following components:

| | |
|---|---|
| `dist1_sq` | N x N matrix; contains values of pairwise squared distances in the x-coordinate. |
| `dist2_sq` | N x N matrix; contains values of pairwise squared distances in the y-coordinate. |
| `dist12` | N x N matrix; contains values of pairwise signed cross- distances between the x- and y-coordinates. |
| `scale_factor` | Value of the scale factor used to rescale distances. |

## Examples

```
# TODO
```

---

`nsDist3d` *Calculate coordinate-specific distance matrices, only for nearest neighbors and store in an array*

---

## Description

`nsDist3d` generates and returns new 3-dimensional arrays containing the former dist1_sq, dist2_s1, and dist12 matrices, but only as needed for the k nearest-neighbors of each location. these 3D matrices (dist1_3d, dist2_3d, and dist12_3d) are used in the new implementation of calculateAD_ns().

## Usage

```
nsDist3d(coords, nID, scale_factor = NULL, isotropic = FALSE)
```

## Arguments

| | |
|---|---|
| `coords` | N x 2 matrix; contains the x-y coordinates of stations. |
| `nID` | N x k matrix; contains indices of nearest neighbors. |
| `scale_factor` | Scalar; optional argument for re-scaling the distances. |
| `isotropic` | Logical; indicates whether distances should be calculated separately for each coordinate dimension (FALSE) or simultaneously for all coordinate dimensions (TRUE). `isotropic = TRUE` can only be used for two-dimensional coordinate systems. |

## Value

Arrays with nearest neighbor distances in each coordinate direction.

## Examples

```
# TODO
```

---

`NSGP-class` *NSGP class*

---

## Description

TODO: more detailed description here

## Fields

`A` A is a field

---

`nsgpModel` *NIMBLE code for a generic nonstationary GP model*

---

## Description

TODO: add documentation

## Usage

```
nsgpModel(tau_model = "constant", sigma_model = "constant",
  Sigma_model = "constant", mu_model = "constant",
  likelihood = "fullGP", returnModelComponents = FALSE,
  constants = list(), z, ...)
```

## Arguments

| | |
|---|---|
| `tau_model` | Character; specifies the model to be used for the log(tau) process. Options are `"logLinReg"` (log-linear regression), `"mixComp"` (mixture component representation), `"GP"` (stationary Gaussian process), and `"approxGP"` (approximation to a Gaussian process). |
| `sigma_model` | Character; specifies the model to be used for the log(sigma) process. See `tau_model` for options. |
| `Sigma_model` | Character; specifies the model to be used for the Sigma anisotropy process. Options are `"covReg"` (covariance regression), `"compReg"` (componentwise regression), `"npMixComp"` (nonparameteric regression via the mixture component approach), `"npGP"` (nonparameteric regression via a stationary Gaussian process), or `"npApproxGP"` (nonparameteric regression via an approximation to a stationary Gaussian process). |
| `mu_model` | TODO |
| `likelihood` | TODO |
| `returnModelComponents` | TODO |
| `constants` | TODO |
| `z` | TODO |
| `...` | TODO |

## Value

A `nimbleCode` object.

## Examples

```
# TODO
```

---

`orderCoordinatesMMD` *Order coordinates according to a maximum-minimum distance criterion.*

---

## Description

`orderCoordinatesMMD` orders an array of (x,y) spatial coordinates according to the "maximum minimum distance" (MMD), as described in Guinness, 2018. (Points are selected to maximize their minimum distance to already- selected points).

## Usage

```
orderCoordinatesMMD(s, exact = FALSE)
```

## Arguments

| | |
|---|---|
| `s` | N x 2 array of N 2-dimensional (x,y) spatial coordinates. |
| `exact` | Logical; `FALSE` uses a fast approximation to MMD ordering (and is almost always recommended), while `TRUE` uses exact MMD ordering but is infeasible for large number of locations. |

## Value

A list with two components: (1) an N x 2 array containing the same spatial coordinates, ordered by MMD, and (2) the same thing, but with any NA values removed.

## Examples

```
# TODO
```

## sgvSetup — *One-time setup wrapper function for the SGV approximation*

### Description

`sgvSetup` is a wrapper function that sets up the SGV approximation. Three objects are required: (1) ordering the locations, (2) identify nearest neighbors, and (3) determine the conditioning set. This function only needs to be run once per SGV analysis.

### Usage

```
sgvSetup(locs, locs_pred = NULL, k = 15, seed = NULL)
```

### Arguments

| | |
|---|---|
| `locs` | Matrix of observed locations. |
| `locs_pred` | Optional matrix of prediction locations. |
| `k` | Number of neighbors. |
| `seed` | TODO |

### Value

A list with the following components:

| | |
|---|---|
| `ord` | A vector of ordering position for the observed locations. |
| `ord_pred` | A vector of ordering position for the prediction locations (if `locs_pred` is provided). |
| `ord_all` | A concatenated vector of `ord` and `ord_pred`. |
| `locs_ord` | A matrix of ordered locations (observed and prediction), included for convenience. |
| `nID_ord` | A matrix of (ordered) neighbor indices. |
| `condition_on_y_ord` | A matrix indicating whether the conditioning set for each (ordered) location is on the latent process (y, `1`) or the observed values (z, `0`). |

### Examples

```
# TODO
```

# Index