Package 'BayesianOHC'

September 6, 2021

Type Package

Title Non-stationary Gaussian Processes for Modeling Ocean Heat Content

Version 0.1.0

Maintainer Samuel Baugh <samuelbaugh@ucla.edu>

Description Code to for fitting non-stationary Gaussian processes to spherical data where a cylindrical representation can be used; intended for quantying ocean heat content from Argo floats however most functions are general and can be applied to other datasets, particularly on the sphere. Non-stationarity is achieved in each parameter through kernel convolutions, where the parameters are allowed to vary flexibly over space. Gaussian hyper-prior surfaces are used for the representation of the parameter fields. Any of the parameter fields can be restricted to stationarity if desired. Code is provided for computing cross-validation with two different metrics; the first computes validation scores by excluding a window of obversations around each point. The second, leave-one-float-out cross-validation, is intended for Argo data.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports ggplot2,
GPvecchia,
maptools,
Matrix,
methods,
RColorBrewer,
sf,
tmap,
verification

Depends R (>= 2.10)

R topics documented:

`add_3d_coord` *Adds 3D coordinates to dataframe*

Description

Adds 3D coordinates to dataframe

Usage

```
add_3d_coord(mydf, R = 1)
```

Arguments

`mydf`	Input dataframe
`R`	Radius of the earth; defaults to unit circle but uses R=6361 if "earth" is used as the argument

Value

Returns mydf with coord1, coord2, and coord3 added

add_means *Add means from GpGp model fit back to data*

Description

Add means from GpGp model fit back to data

Usage

```
add_means(yeardata, model_fit)
```

Arguments

yeardata	Data from a particular year
model_fit	GpGp model fit containing design matrix and parameters

augment_data *Augments data with parameter values*

Description

Adds parameters to a given data frame by kriging the basis values specified in "myparams" to the locations in "mydata"

Usage

```
augment_data(
  pred_locs,
  myparams,
  varname_list = NULL,
  linkfuns = default_linkfuns
)
```

Arguments

pred_locs	Locations for kriging the parameter fields
myparams	Basis parameter values for kriging
varname_list	List of parameter fields to krig
linkfuns	List that contains an entry for each parameter field specified in varname_list

Value

Returns a data frame that is a copy of the input data but with new parameter values for each of the fields specified in varname_list

`augment_data_parallel` *Parallel version of augment_data*

Description

Adds parameters to a given data frame by kriging the basis values specified in "myparams" to the locations in "mydata"

Usage

```
augment_data_parallel(
  pred_locs,
  myparams,
  varname_list = NULL,
  linkfuns = default_linkfuns,
  ncores = 1
)
```

Arguments

`pred_locs`	Locations for kriging the parameter fields
`myparams`	Basis parameter values for kriging
`varname_list`	List of parameter fields to krig
`linkfuns`	List that contains an entry for each parameter field specified in varname_list
`ncores`	If ncores>1 parallelization is used

Value

Returns a data frame that is a copy of the input data but with new parameter values for each of the fields specified in varname_list

`build_veccmat_list_grouped` *Computes list of cholesky factors of the Vecchia precision matrix*

Description

Computes list of cholesky factors of the Vecchia precision matrix

Usage

```
build_veccmat_list_grouped(
  augdata_ordered,
  grouping_list,
  yearlist = 2007:2016,
  corrfun = cyl_cor_single,
  ncores = 1,
  verb = F
)
```

Arguments

augdata_ordered	Ordered data over which to compute the factors. Data should be 'augmented' meaning that each row has the values of the kernel parametmers for that locations; this can be added to a dataframe by calling "augment_data" with a parameter object
grouping_list	The list of grouping information lists for the process
yearlist	The list of years for creating the factors
corrfun	Correlation function to use, default is cyl_cor_single
ncores	If ncores>1, parallelization is used
verb	Depreciated

Value

Returns a list of cholesky factors of the Vecchia precision matrix corresponding to the years in yearlist

---

chol_lik *Cholesky likelihood function*

Description

Cholesky likelihood function

Usage

```
chol_lik(z, ranges, locs, corfun)
```

Arguments

z	Simulated observed values
ranges	Vector of range values
locs	Locations
corfun	Function for computing correlations

chord_cor_double *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
chord_cor_double(ii, jj, input1, input2)
```

Arguments

ii	First index
jj	Second index
input1	First location dataframe (needs coord1,coord2,coord3)
input2	Second location dataframe (needs coord1,coord2,coord3)

chord_cor_exp_double *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
chord_cor_exp_double(ii, jj, input1, input2)
```

Arguments

ii	First index
jj	Second index
input1	First location dataframe (needs coord1,coord2,coord3)
input2	Second location dataframe (needs coord1,coord2,coord3)

chord_cor_exp_single *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
chord_cor_exp_single(ii, jj, myinput)
```

Arguments

ii	First index
jj	Second index
myinput	Location dataframe (needs coord1,coord2,coord3)

chord_cor_single *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
chord_cor_single(ii, jj, myinput)
```

Arguments

ii	First index
jj	Second index
myinput	First dataframe

compute_cv_df *Computes cross-validation scores and standard errors*

Description

Computes cross-validation scores and their associated prediction errors using either lofo (leave-one-float-out) or lowo (leave-one-window-out) cross validation methods

Usage

```
compute_cv_df(
  augdata_ordered,
  cv_params,
  cv_function = cv_scores,
  cv_type = "lofo",
  yearlist = 2007:2016,
  nsamp = -1,
  ncores = 1,
  cv_options = NULL
)
```

Arguments

augdata_ordered	The input data augmented with the kernel parameter values at each location (can be created with function "augment_data")
cv_params	List of parameters to the cross-validation function; should be the same length as yearlist. If cvtype=standard this is a list of veccmat objects, if cvtype=levitus this should be a distance matrix between each of the observations in each year in kilometers, and if cvtype=gpgp this should be a fitted GpGp model
cv_function	Default is "cv_scores", an availabel option is "levitus_preds".
cv_type	lofo (leave-one-float-out) vs lowo (leave-one-window-out)
yearlist	The list of years over which to compute the validation scores
nsamp	If nsamp=-1 computes scores for all observation locations; if not validation scores are computed for a selection of nsamp indices for each year
ncores	If ncores>1, how many cores should be used for parallelization
cv_options	Options for cv; if lowo is chosen you can input the width of the window to withold

Value

Returns a dataframe the same size of "augdata_ordered" with two additional columns, validval and validsd, giving the mean and standard deviation of the leave-one-float-out validation score at that point

`compute_grouped_conditioning_sets` *Computes grouped conditioning sets for use in the Vecchia approximation.*

Description

Computes grouped conditioning sets for use in the Vecchia approximation.

Usage

```
compute_grouped_conditioning_sets(NNarray, trivial = F)
```

Arguments

`NNarray`	Matrix containing indices of nearest neighbors; generally returned from compute_nnarray
`trivial`	If true returns grouping object with no grouping done

`compute_grouping_list` *Computes list of grouped conditioning sets for each year in the given data*

Description

Computes list of grouped conditioning sets for each year in the given data

Usage

```
compute_grouping_list(locations, m, yearlist = 2007:2016, verb = F, ncores = 1)
```

Arguments

`locations`	Ordered data on which to compute nearest neighbors
`m`	Number of nearest neighbors to use in constructing the ungrouped vecchia conditioning sets.
`yearlist`	List of years over which to compute the group sets.
`verb`	T/F, if T displays a progress bar
`ncores`	Number of cores to use, if ncores>1 parallelization is

Value

Returns a list of length yearlist where each entry contains the output of compute_grouped_conditioning_sets on the data for the corresponding year.

`compute_lincomb_predictions` *Compute kriging distributions for linear combinations*

Description

Computes posterior distribution for a linear combination of predicted values

Usage

```
compute_lincomb_predictions(
  obsdata,
  pred_locs,
  myparams,
  grouping_list_preddf,
  varname_list = default_varname_list,
  scalarvec = NULL,
  yearlist = 2007:2016,
  ncores = 1,
  ret_var = T
)
```

Arguments

`obsdata`	Data containing observations
`pred_locs`	Locations for predictions
`myparams`	Parameter object for kriging
`grouping_list_preddf`	List of group information for each year
`varname_list`	List of varnames to use
`scalarvec`	Vector of scalars for linear combination. If missing, this will be inferred by assuming pred_locs is a grid
`yearlist`	Years to use in kriging
`ncores`	If ncores>1 parallelization is used
`ret_var`	T/F, if T returns the kriging variance of the linear combination as well as the mean

Value

Returns the value of pred of kriging predictions located at pred_locs

`compute_nnarray` *Computes array of nearest neighbors to be used for Vecchia process conditioning sets*

Description

Computes array of nearest neighbors to be used for Vecchia process conditioning sets

Usage

```
compute_nnarray(locations, m, verb = F, ncores = 1)
```

Arguments

`locations`	Ordered data on which to compute nearest neighbors
`m`	Number of nearest neighbors
`verb`	T/F, if T displays a progress bar
`ncores`	Number of cores to use, if ncores>1 parallelization is used

Value

Returns a matrix of size (nlocs,m) where nlocs is the number of locations specified in "ordered_input".

`compute_nn_clusters` *Finds clusters of sets of nearest neighbors for grouping sets. Function used internally and is adapted from GpGP package (see in-function comments for citation details).*

Description

Finds clusters of sets of nearest neighbors for grouping sets. Function used internally and is adapted from GpGP package (see in-function comments for citation details).

Usage

```
compute_nn_clusters(NNarray)
```

Arguments

`NNarray`	Matrix continaing indices of nearest neighbors; generally returned from compute_nnarray

Value

Returns a list of clustered arrays of nearest neighbors of length m, where m is the second dimension of NNarray

`convert_gpgp_fit_to_cvparams` *Convert GpGp model fit object to veccmat object*

Description

Convert GpGp model fit object to veccmat object

Usage

```
convert_gpgp_fit_to_cvparams(
  gpgp_fit_list,
  ordered_data,
  corrfun,
  yearlist = 2007:2016
)
```

Arguments

`gpgp_fit_list`	List of model fits GpGp::fit_model
`ordered_data`	Observation data
`corrfun`	Correlation function to use
`yearlist`	Lits of years to use

Value

Returns a list of "veccmat" objects

`convert_nsgp_to_cvparams` *Function to convert BayesNSGP samples to cv_params object*

Description

Function to convert BayesNSGP samples to cv_params object

Usage

```
convert_nsgp_to_cvparams(
  nsgp_vals,
  knot_points,
  argo_data_chord,
  grouping_list
)
```

Arguments

knot_points	Knot points used in model fit
argo_data_chord	Argo data with 3D euclidean coordinates
grouping_list	List of groups to use
nspg_vals	Samples obtained from fitting a BayesNSGP model

---

convert_params_to_statiso

Convert nonstationary anisotropic parameters to stationary or isotropic

---

Description

Convert nonstationary anisotropic parameters to stationary or isotropic

Usage

```
convert_params_to_statiso(myparams, myaugdata, mymodel)
```

Arguments

myparams	Parameters to convert
myaugdata	Augmented data to use in the conversion
mymodel	Contains list of parameters to convert to stationarity and a boolean "iso" for whether or not the model should be converted to isotropic

---

convert_theta_lat_to_effective_range_deg

Converts theta lat to effective range in degrees

Description

Converts theta lat to effective range in degrees

Usage

```
convert_theta_lat_to_effective_range_deg(thetalat)
```

Arguments

thetalat	Theta lat

convert_theta_lon_to_effective_range_deg

Converts theta lon to effective range in degrees

Description

Converts theta lon to effective range in degrees

Usage

```
convert_theta_lon_to_effective_range_deg(thetalon)
```

Arguments

thetalon	Theta lon

create_veccmat_grouped

Creat grouped veccmat objects

Description

Computes cholesky factors of the Vecchia precision matrix corresponding to a specified grouping of conditioning sets. Code inspired by similar code in GPVecchia package, see source code for citation.

Usage

```
create_veccmat_grouped(
  augdata_ordered,
  groupobj,
  ncores = 1,
  corrfun = cyl_cor_single,
  covparms = NULL
)
```

Arguments

augdata_ordered	Ordered data over which to compute the factors. Data should be 'augmented' meaning that each row has the values of the kernel parametmers for that locations; this can be added to a dataframe by calling "augment_data" with a parameter object
groupobj	The grouping information for the Vecchia process
ncores	Number of cores to use, if ncores>1 then parallelization

corrfun	Which function to use for correlations; default cylindrical is used
covparms	Optional, only needed if stationary corrfun is used. For other correlation functions the non-stationary parameters for the kernel convolutions should be included the augdata_ordered dataframe

---

cv_scores *Compute cross-validation scores with veccmat object*

Description

Compute cross-validation scores with veccmat object

Usage

```
cv_scores(cv_indices, obspred_df, veccmat)
```

Arguments

cv_indices	Indices to predict in cross-validation
obspred_df	Dataframe containing both observation and prediction locations
veccmat	Veccmat object corresponding to data in obspred_df

---

cylindrical_correlation_exact *Computes cylindrical correlations using kernel convolutions*

Description

Computes the exact longitudinal correlation kernel convolution between locations loc1 and loc2 with longitudinal ranges range1 and range2 respectively. Since this is over a circular domain Gaussian error function calls are required.

Usage

```
cylindrical_correlation_exact(loc1, loc2, range1, range2)
```

Arguments

loc1	The first location on the longitudinal circle
loc2	The second location on the longitudinal circle
range1	The longitudinal range for the first location
range2	The longitudinal range for the second location

Value

Returns the value of the exact cylndrical correlation; see supplementary material for details

cylindrical_correlation_gaussian

Cylindrical correlations using Gaussian approximation

Description

Computes the approximate longitudinal correlation kernel convolution between locations loc1 and loc2 with longitudinal ranges range1 and range2 respectively.

Usage

```
cylindrical_correlation_gaussian(loc1, loc2, range1, range2)
```

Arguments

loc1	The first location on the longitudinal circle
loc2	The second location on the longitudinal circle
range1	The longitudinal range for the first location
range2	The longitudinal range for the second location Returns the Gaussian approximation to the full convolution, which is accurate if the effective range is less than about 100 degrees; see supplementary material for details.

cylind_approx_simulation

Runs simulation comparing exact cylindrical convolutions with Gaussian approximation

Description

Runs simulation comparing exact cylindrical convolutions with Gaussian approximation

Usage

```
cylind_approx_simulation(true_range_seq, rangefun, nrep, locs)
```

Arguments

true_range_seq	Sequence of true range parameters
rangefun	Function for computing non-stationary range parameters
nrep	Number of repetitions to simulate data
locs	Locations on which to generate data

cyl_cor_convolution_single *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
cyl_cor_convolution_single(ii, jj, myinput)
```

Arguments

ii	First index
jj	Second index
myinput	First dataframe

cyl_cor_double *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
cyl_cor_double(ii, jj, input1, input2)
```

Arguments

ii	First index
jj	Second index
input1	First dataframe
input2	Second dataframe

cyl_cor_exact_double *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
cyl_cor_exact_double(ii, jj, input1, input2)
```

Arguments

ii	First index
jj	Second index
input1	First dataframe
input2	Second dataframe

cyl_cor_exact_single *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
cyl_cor_exact_single(ii, jj, myinput)
```

Arguments

ii	First index
jj	Second index
myinput	First dataframe

cyl_cor_single *Helper function for computing correlations*

Description

Helper function for computing correlations

Usage

```
cyl_cor_single(ii, jj, myinput)
```

Arguments

ii	First index
jj	Second index
myinput	First dataframe

euclidean_correlation_convolution *Computes euclidean correlations using kernel convolutions*

Description

Computes the exact latitudinal correlation kernel convolution between locations loc1 and loc2 with latitudinal ranges range1 and range2 respectively

Usage

```
euclidean_correlation_convolution(loc1, loc2, range1, range2)
```

Arguments

loc1	The first location on the longitudinal circle
loc2	The second location on the longitudinal circle
range1	The longitudinal range for the first location
range2	The longitudinal range for the second location

Value

Returns the exact value of of the convolution in the Euclidean (latitudinal) dimension

gaussian_integral *Internal function for computing exact kernel convolutions over the cylindrical dimension.*

Description

Internal function for computing exact kernel convolutions over the cylindrical dimension.

Usage

```
gaussian_integral(loc1, loc2, range1, range2, upper, lower)
```

Arguments

loc1	First location
loc2	Second location
range1	Range parameter for first location
range2	Range parameter for second location
upper	Upper bound
lower	Lower bound

gen_deltas_from_grid *Generates delta values on a grid for computing numerical integrals*

Description

Generates delta values on a grid for computing numerical integrals

Usage

```
gen_deltas_from_grid(grid_locs, res = NULL)
```

Arguments

grid_locs	Grid locations
res	Resolution of the grid; if not included will be inferred

`gen_masked_grid` *Generates grid over the sphere using the mask saved in MCMC_Input/mask.RData (for our purposes this mask is obtained from the Roemmich-Gilson climatology)*

Description

Generates grid over the sphere using the mask saved in MCMC_Input/mask.RData (for our purposes this mask is obtained from the Roemmich-Gilson climatology)

Usage

```
gen_masked_grid(latres = 1, lonres = 1)
```

Arguments

`latres`	The latitudinal resolution of the desired grid
`lonres`	The longitudinal resolution of the desired grid

Value

Returns a dataframe with columns "lat_rad", "lon_rad", "lat_degrees", and "lon_degrees" corresponding to the gridpoint locations

`get_crps` *Computes CRPS values*

Description

Computes CRPS values

Usage

```
get_crps(x)
```

Arguments

`x`	Validation dataframe

get_mu_integrated_posterior

Computes mean/trend integrated posteriors

Description

Computes the mean and variance for the posterior distribution of the globally integrated mean and trend fields as well as their correlation

Usage

```
get_mu_integrated_posterior(sampleout, myparams, pred_locs)
```

Arguments

sampleout	The output of sample_meanfield
myparams	The current iteration's parameter object
pred_locs	Grid for computing integrated field

Value

Returns a list containing mu0 (the posterior mean of the integrated mean field), mu0_var (the posterior variance of the integrated mean field), slope and slope_var analogously, and corr_term which gives the correlation between the integrated mean and slopes

get_region_params

For basis parameter, restricts the knot locations to a specific region and re-krigs the parameter values.

Description

For basis parameter, restricts the knot locations to a specific region and re-krigs the parameter values.

Usage

```
get_region_params(
  inputparams,
  region_name,
  varname_list = default_varname_list,
  invlinkfuns = default_invlinkfuns
)
```

Arguments

inputparams	Basis function parameters
region_name	Region name, see documentation for "in_region" for available options
varname_list	List of names of parameters to convert
invlinkfuns	Inverse link functions for parameter fields

Value

Returns a list of parameters with the same hyperparameters as inputparameters

---

get_rmse *Computes RMSE values*

Description

Computes RMSE values

Usage

```
get_rmse(x)
```

Arguments

x	Validation dataframe

---

get_stationary_MLEs *Find MLEs for stationary process*

Description

Computes maximum likelihood estimates for an isotropic cylindrical model

Usage

```
get_stationary_MLEs(
  mydata,
  yearlist = 2007:2016,
  verb = F,
  prior_list = NULL,
  est_nugget = F,
  est_mu = T,
  est_slope = T,
  iso = F
)
```

Arguments

mydata	dataframe with column z containing osbervations
yearlist	list of years to use, default is 2007:2016
verb	If T use trace=6 in optim
prior_list	The list of coefficients for normal priors on the log variables (if NULL or missing no priors are used)
est_nugget	T/F if T nugget is estimated
est_mu	T/F, if T constant mean is estimated
est_slope	T/F, if T linear slope is estimated
iso	T/F, if T an isotropic model will be used

Value

Returns a list containing the maximum likelihood estimates for a stationary cylindrical model on the given data. Output contains "theta_lat", "theta_lon", "nugget", "mu0", "phi", "slope", "likelihood", "data_mean" and "data_var"

---

gp_profiled_stationary_likelihood

Computes profiled log-likelihood for stationary Gaussian process

---

Description

Compute Gaussian process log-likelihood with options to profile over the mean and variance parameters

Usage

```
gp_profiled_stationary_likelihood(
  mydata,
  logparams,
  yearlist = 2007:2016,
  return_profiled = F,
  prior_list = NULL,
  est_mu = F,
  est_slope = F,
  iso = F
)
```

Arguments

mydata	dataframe with column z containing osbervations
logparams	vector of log parameters (theta_lat,theta_lon,nugget)
yearlist	list of years to use, default is 2007:2016
return_profiled	T/F, if T profiled parameters are returned
prior_list	The list of coefficients for normal priors on the log variables (if NULL or missing no priors are used)
est_mu	T/F, if T constant mean is estimated
est_slope	T/F, if T linear slope is estimated
iso	T/F, if T an isotropic model will be used

Value

If return_profiled=F, returns the log-likelihood. Otherwise, returns a list containing the log-likelihood and the profiled mu0, slope, and phi (variance) values

---

groupinfo_full_chol *Computes grouping object for full Cholesky*

Description

Computes grouping object for full Cholesky

Usage

```
groupinfo_full_chol(nobs)
```

Arguments

nobs	Number of observations

in_region *Restricts the dataframe given to a specified region.*

Description

Restricts the dataframe given to a specified region.

Usage

```
in_region(df, region_name)
```

Arguments

df	Dataframe containing locations in columns "lon_degrees" and "lat_degrees".
region_name	Options are North Atlantic (AtlN), Pacific Northeast (PacNE), Pacific Northwest (PacNW), Tropical Atlantic (AtlTrop), Western Tropical Pacific (PacTropW), Eastern Tropical Pacific (PacTropE), Indian (Ind), Southern Pacific (PacSo), Southern Indian (IndSo), Southern Atlantic (AtlSo), and globe. See manuscript or source code for region definitions

Value

Returns a subset of df with locations constrained to the specified region.

krig_argo_field *Computes kriging values and optionally variances*

Description

Computes kriging values and optionally variances

Usage

```
krig_argo_field(input_data, pred_indices, ret_var = F)
```

Arguments

input_data	Data containing both observation locations and prediction locations
pred_indices	Indices of locations in the input_data object on which predictions should be computed; it is implied that indicies not in this vector are observation locations.
ret_var	T/F, if T compute and return the kriging variane

Value

Returns a list with entries "val" containing the kriged values at the prediction locations, and "var" containing the kriging variances (only if ret_var=T)

`krig_basis_to_field` *Creates parameter field from basis values*

Description

Creates parameter field from basis values

Usage

```
krig_basis_to_field(
  myparams,
  predlocs,
  varname,
  linkfun,
  corrfun = cyl_cor_double
)
```

Arguments

`myparams`	List containing knot locations and basis values
`predlocs`	Data frame containing locations for kriging
`varname`	Name of parameter field to krig
`linkfun`	Link function for field
`corrfun`	Correlation function to use

Value

Returns a vector of parameters at the locations specificed by predlocs

`levitus_preds` *Compute predictions using method described by Levitus et al. (2012), see supplementary materials for details*

Description

Compute predictions using method described by Levitus et al. (2012), see supplementary materials for details

Usage

```
levitus_preds(
  pred_indices,
  obspred_df,
  distmat_km = NULL,
  distfun = NULL,
  verb = F
)
```

Arguments

pred_indices	Indices to predict
obspred_df	Dataframe containing observation and prediction locations
distmat_km	Optional, can make it faster if doing cross validation than having to re-compute distances
distfun	If distmat not supplied, function for computing distances; should be geodist for great circle distances
verb	Should progress be displayed?

Value

Dataframe with "validval" (predictions) and "validsd" (standard errors)

---

mygc_dist *Compute great circle distance in radians*

Description

Compute great circle distance in radians

Usage

```
mygc_dist(l1, l2)
```

Arguments

l1	Argument 1
l2	Argument 2

---

mygc_dist_degrees *Compute great circle distance in degrees*

Description

Compute great circle distance in degrees

Usage

```
mygc_dist_degrees(l1, l2)
```

Arguments

l1	Argument 1
l2	Argument 2

order_yeardata *Orders data for each year*

Description

Orders data for each year in "yearlist" according to the max-min distance ordering

Usage

```
order_yeardata(unordered_data, yearlist = 2007:2016)
```

Arguments

unordered_data	Un-ordered data to order. Must contain a "years" column. Data for each year will be ordered separately.
yearlist	List of years to order.

Value

Returns a dataframe the same size as "unordered_data"

paramgrid_to_plots *Creates plots for each variable location on a grid*

Description

Creates plots for each variable location on a grid

Usage

```
paramgrid_to_plots(
  plotgrid,
  configtype = "",
  breaklist,
  which_plots = c("stdev", "nugget", "efflat", "efflon", "mu", "slope")
)
```

Arguments

plotgrid	Gridded datframe containing values to plot
configtype	Configuration type for labels
breaklist	List of breakpoints for each variable
which_plots	Which variables to plot

plot_cut *Plots map of observations "cut" by supplied breaks*

Description

Plots map of observations "cut" by supplied breaks

Usage

```
plot_cut(
  mydf,
  mybreaks,
  varsym,
  varname,
  legendname,
  mytitle = "",
  is_grid = T,
  rounddigit = 1,
  mycex = 1,
  mylabels = NULL
)
```

Arguments

mydf	Dataframe to plot
mybreaks	Breakpoints to use
varsym	Symbol of variablename for legend
varname	Name of variable to plot
legendname	Name of legend
mytitle	Plot title
is_grid	Is data located on grid?
rounddigit	How many digits to round?
mycex	Cex value for plotting if is_grid=F
mylabels	Labels to use in legend

`plot_cut_diverging` *Plots map of observations "cut" by supplied breaks with diverging colorbar*

Description

Plots map of observations "cut" by supplied breaks with diverging colorbar

Usage

```
plot_cut_diverging(
  mydf,
  mybreaks,
  varsym,
  varname,
  legendname,
  mytitle = "",
  is_grid = T,
  rounddigit = 1,
  mycex = 1,
  mylabels = NULL
)
```

Arguments

`mydf`	Dataframe to plot
`mybreaks`	Breakpoints to use
`varsym`	Symbol of variablename for legend
`varname`	Name of variable to plot
`legendname`	Name of legend
`mytitle`	Plot title
`is_grid`	Is data located on grid?
`rounddigit`	How many digits to round?
`mycex`	Cex value for plotting if is_grid=F
`mylabels`	Labels to use in legend

`pred_with_veccmat` *Predictions using veccmat input*

Description

Computes predictions using a cholesky precision matrix for a vecchia process. Inspired by similar code in GPVecchia package; see source code for citation.

Usage

```
pred_with_veccmat(
  augdata_ordered_full,
  pred_indices,
  veccmat,
  obs_indices = !pred_indices,
  ret_var = F,
  scalarvec = NA
)
```

Arguments

`augdata_ordered_full`	Data frame containing both observations and locations as well as prediction locations
`pred_indices`	Specifies which indices of augdata_ord_full are to be predicted
`veccmat`	The cholesky factor of the precision matrix for the Vecchia process to be predicted
`obs_indices`	The indices in augdata_ordered_full that correspond to observations. By default this is the complement of the indices in pred_indices
`ret_var`	T/F, it T calculates and returns the kriging variance
`scalarvec`	If specified will compute predictions and variances for the linear combination scalarvec*preds

Value

If ret_var=F, returns the vector of predicted values at the specified prediction locations. Otherwise returns a list with entries "pred" and "var". If scalarvec is specified returns the prediction and variance for the linear combination.

`record_status` *Records status of mcmc sampler*

Description

Records the status of the mcmc sampler through appending to the file specified in "status_filename".

Usage

```
record_status(
  status_filename,
  iteration,
  current_likelihood,
  current_prior,
  accepts,
  variance_scaling_factor,
  start_time
)
```

Arguments

`status_filename`	Location to write the sampler status.
`iteration`	The current iteration.
`current_likelihood`	The current likelihood.
`current_prior`	The current prior value.
`accepts`	The current vector of Metropolis-Hastings acceptances.
`variance_scaling_factor`	The current vector of scaled proposal variances.
`start_time`	The original start time for the sampler

Value

Returns a data frame that is a copy of the input data but with new parameter values for each of the fields specified in varname_list

resample_slopes *Re-samples slopes from marginal posterior*

Description

Re-samples slopes from the marginal posterior conditional on the values of the other parameters.

Usage

```
resample_slopes(mu_sampleout, nresamp, krig_grid, scalarvec = NULL)
```

Arguments

mu_sampleout	List containing parameters and posterior covariances for the mean and trend fields. Should be the output of "sample_mean_trend" run with return_full_posterior=T.
nresamp	Number of re-sampled fields to return.
krig_grid	Grid for computing integrated mean/trend values.
scalarvec	Values for numerical integrated; if missing will infer from grid.

Value

Returns a list with "trend_samples", a nresample(x)ngrid matrix where each row contains the trend values of a resampled field at the corresponding grid-points, "intslope_samples" which is a vector of length nresamp giving the integrated values of "trend_samples", "intmean" which gives the posterior mean of the integrated trend, and "intvar" which gives the posterior variance of the integrated trend.

run_mcmc_sampler *Runs MCMC sampler for the cylindrical model with linear trend term*

Description

Runs MCMC sampler for the cylindrical model with linear trend term

Usage

```
run_mcmc_sampler(
  ordered_data,
  initparams,
  M = 20000,
  m = "chol",
  yearlist = 2007:2016,
  pred_locs = NULL,
  grouping_list = NULL,
  grouping_list_preddf = NULL,
  outdir = "../MCMC_Output/",
```

```
  varname_list = NULL,
  corrfun = cyl_cor_single,
  var_sample_order = NULL,
  stationary_varnames = NULL,
  region_name = "globe",
  continue_from_previous = F,
  run_label = "mcmc",
  linkfuns = default_linkfuns,
  ret_last_veccmat = F,
  invlinkfuns = default_invlinkfuns,
  pred_iter = Inf,
  save_iter = Inf,
  plot_iter = Inf,
  ncores = 1
)
```

Arguments

ordered_data	Data for running the sampler; should be already ordered.
initparams	Parameters for initializing the sampler.
M	number of iterations to run the sampler.
m	Can either be the string "chol" indicating that the full Cholesky should be used, or an integer indicating the number of Vecchia neighbors that should be used.
yearlist	The list of years to be used.
pred_locs	Locations to compute predictions
grouping_list	List of observation location groupings for Vecchia approximation, if not included the list will be computed using the value of m.
grouping_list_preddf	List of groupings of prediction locations (as specified in pred_locs); if empty this will be computed in the function
outdir	Path to directory for storing output.
varname_list	List of variable names to sample.
corrfun	Correlation fun to use, defaults to cylindrical
var_sample_order	Order of variables names for sampling. Note that mu0 and slope should be specified as "mu" in this list as they are sampled together from their joint marginal posterior distribution
stationary_varnames	List containing which of the specified variables should be maintained as stationary.
region_name	Name of the region to restrict to; see documentation of "in_region" for available options. Data does not need to be restricted to the region beforehand.
continue_from_previous	T/F, if T continue from previous iteration.

run_label	Label used in saving the output. If label is reused and continue_from_previous=F then the previous run will be overwritten
linkfuns	List of link functions for the variables specified in varname_list.
ret_last_veccmat	T/F, if T returns the last veccmat list
invlinkfuns	Inverse link functions for the variables specified in varname list.
pred_iter	If do_predictions=T the sampler will compute predictions for every pred_iter iterations
save_iter	Every save_iter iterations, the samples since the last checkpoint will be saved in the output directory. Default is to not save samples and rather return all of the samples at the end.
plot_iter	At each plot_iter iterations figures showing the parameter field maps will be saved in the output directory (note: plotting not currently implemented)
ncores	Number of cores to be used; if ncores>1 then parallelization will be used

Value

Returns the last M-save_iter samples, or all M samples if save_iter>M. If save_iter<M, earlier samples are stored every save_iter iterations in the directory yspecified by outdir

---

sample_mean_trend *Sample mean and trend fields*

Description

Samples the mean and trend parameters from their joint posterior distribution conditioned on the other parameter fields

Usage

```
sample_mean_trend(
  augdata_ordered,
  myparams,
  veccmat_list,
  yearlist = c(2007:2016),
  return_posterior_mean = T,
  return_full_posterior = F,
  ncores = 1,
  seed = "rand"
)
```

Arguments

augdata_ordered	The data from which to compute the posterior.
myparams	The basis values for the other parameter fields; also needs to contain hyperpriors for the mean and trend fields.
veccmat_list	List of cholesky factors of precision matrices for each year in yearlist.
yearlist	List of years over which to compute the posterior
return_posterior_mean	T/F, if T returns the posterior mean, if not returns a sample from the posterior distribution.
return_full_posterior	T/F, if T returns the covariance matrix between the mean and trend fields
ncores	If ncores>1, parallelization is used
seed	Default is 'rand' which uses a random seed, otherwise sets a user-defined seed

Value

If return_full_posterior=F returns a parameter object with the updated basis values for the mean and slope fields. If return_full_posterior=T, returns a list containing the new parameters as well as the mean and covariance matrix for the posterior distribution.

---

save_image *Saves high-quality image and trims white space from result*

Description

Saves high-quality image and trims white space from result

Usage

```
save_image(image, file)
```

Arguments

image	Image to save
file	Filename

simulate_mles *Simulates data using true parameters, and finds MLEs*

Description

MLEs are found using both cylindrical_correlation_exact anad cylindrical_correlation_exact

Usage

```
simulate_mles(true_ranges, rangefun, locs)
```

Arguments

true_ranges	The true ranges to use for the simulation
rangefun	Function for computing non-stationary range parameters
locs	Locations on which to generate data

vecc_lik_from_veccmat *Vecchia likelihood from veccmat object*

Description

Computes the vecchia likelihood from the cholesky of the precision matrix of a vecchia process. Based off of code with similar functionality in the GPVecchia package, see source code for citation details.

Usage

```
vecc_lik_from_veccmat(likinput_ordered, veccmat)
```

Arguments

likinput_ordered	The ordered data on which to evaluate the likelihood, where observational values are stored in the z column
veccmat	The cholesky of the precision matrix corresponding to a vecchia process on the data

Value

Returns the log-likelihood of the observations

`vecc_lik_over_years` *Evaluates the vecchia likelihood over multiple years*

Description

Evaluates the vecchia likelihood over multiple years

Usage

```
vecc_lik_over_years(likinput_ordered, veccmat_list, yearlist = 2007:2016)
```

Arguments

`likinput_ordered`	The ordered data on which to evaluate the likelihood, where observational values are stored in the z column
`veccmat_list`	List of cholesky of the precision matrices corresponding to the years specified in yearlist
`yearlist`	The years over which to calculate the likelihoods

Value

Returns a list of log-likelihoods the same length as yearlist

Index