Package: meshed (via r-universe)

October 17, 2024

Type Package

Title Bayesian Regression with Meshed Gaussian Processes

Version 0.2.3

Date 2022-09-19

Author Michele Peruzzi

Maintainer Michele Peruzzi <michele.peruzzi@duke.edu>

Description Fits Bayesian regression models based on latent Meshed Gaussian Processes (MGP) as described in Peruzzi, Banerjee, Finley (2020) <doi:10.1080/01621459.2020.1833889>, Peruzzi, Banerjee, Dunson, and Finley (2021) <arXiv:2101.03579>, Peruzzi and Dunson (2022) <arXiv:2201.10080>. Funded by ERC grant 856506 and NIH grant R01ES028804.

License GPL (>= 3)

Imports Rcpp (>= 1.0.5), stats, dplyr, glue, rlang, magrittr, FNN

Suggests ggplot2, abind, rmarkdown, knitr, tidyr

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-09-19 22:56:15 UTC

Contents

meshed-package — *Methods for fitting models based on Meshed Gaussian Processes (MGPs)*

Description

meshed is a flexible package for Bayesian regression analysis on spatial or spatiotemporal datasets. The main function for fitting regression models is spmeshed, which outputs posterior samples obtained from Markov chain Monte Carlo which can be summarised using standard tools. The package also provides a function rmeshedgp for quickly simulating correlated spatial or spatiotemporal data at a very large number of locations.

Details

The functions rmeshedgp and spmeshed are provided for prior and posterior sampling (respectively) of Bayesian spatial or spatiotemporal multivariate regression models based on Meshed Gaussian Processes as introduced by Peruzzi, Banerjee, and Finley (2020). Posterior sampling via spmeshed proceeds by default via GriPS as detailed in Peruzzi, Banerjee, Dunson, and Finley (2021). When at least one outcome is not modeled with Gaussian errors, sampling proceeds taking advantage of Metropolis-adjusted Langevin dynamics as detailed in Peruzzi and Dunson (2022).

Author(s)

Michele Peruzzi

References

Peruzzi, M., Banerjee, S., and Finley, A.O. (2022) Highly Scalable Bayesian Geostatistical Modeling via Meshed Gaussian Processes on Partitioned Domains. *Journal of the American Statistical Association*, 117(538):969-982. doi:10.1080/01621459.2020.1833889

Peruzzi, M., Banerjee, S., Dunson, D.B., and Finley, A.O. (2021) Grid-Parametrize-Split (GriPS) for Improved Scalable Inference in Spatial Big Data Analysis. https://arxiv.org/abs/2101.03579

Peruzzi, M., Dunson, D.B. (2022) Spatial meshing for general Bayesian multivariate models. https://arxiv.org/abs/2201.10080

See Also

spmeshed, rmeshedgp

`predict.spmeshed` *Posterior predictive sampling for models based on MGPs*

Description

Sample from the posterior predictive distribution of the outcomes at new spatial or spatiotemporal locations after MCMC.

Usage

```
## S3 method for class 'spmeshed'
predict(object, newx, newcoords,
    n_threads=4, verbose=FALSE, ...)
```

Arguments

`object`	Object output from `spmeshed` with option `settings$saving=TRUE`.
`newx`	matrix of covariate values at the new coordinates.
`newcoords`	matrix of new coordinates.
`n_threads`	integer number of OpenMP threads. This is ineffective if `meshed` was not compiled with OpenMP support.
`verbose`	boolean for progress messagging.
`...`	other arguments (unused).

Details

While this function can always be used to make predictions, in most cases it is more efficient to just include the prediction locations in the main data as `NA` values; `spmeshed` will sample from the posterior predictive distribution at those locations while doing MCMC. The `predict` method is only recommended when all 4 of the following are true:

(1) `spmeshed` was run with `settings$forced_grid=FALSE` and

(2) the prediction locations are uniformly scattered on the domain (or rather, they are not clustered as a large empty area) and

(3) the number of prediction locations is a large portion of the number of observed data points and

(4) the prediction locations are not on a grid.

In all other cases the main `spmeshed` function is setup to be more efficient in automatically performing predictions during MCMC.

Value

`coords_out`	matrix with the prediction location coordinates (order updated after predictions).
`preds_out`	array of dimension (n_o, q, m) where n_o is the number of prediction locations, q is the output dimension, m is the number of MCMC samples.

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>

References

Peruzzi, M., Banerjee, S., and Finley, A.O. (2020) Highly Scalable Bayesian Geostatistical Modeling via Meshed Gaussian Processes on Partitioned Domains. *Journal of the American Statistical Association*, in press. doi:10.1080/01621459.2020.1833889

Peruzzi, M., Banerjee, S., Dunson, D.B., and Finley, A.O. (2021) Grid-Parametrize-Split (GriPS) for Improved Scalable Inference in Spatial Big Data Analysis. https://arxiv.org/abs/2101.03579

Examples

```
# toy example with tiny dataset and short MCMC
# on a univariate outcome
library(magrittr)
library(dplyr)
library(meshed)

set.seed(2021)

SS <- 12
n <- SS^2 # total n. locations, including missing ones

coords <- data.frame(Var1=runif(n), Var2=runif(n)) %>%
  as.matrix()

# generate data
sigmasq <- 2.3
phi <- 6
tausq <- .1
B <- c(-1,.5,1)

CC <- sigmasq * exp(-phi * as.matrix(dist(coords)))
LC <- t(chol(CC))
w <- LC %*% rnorm(n)
p <- length(B)
X <- rnorm(n * p) %>% matrix(ncol=p)
y_full <- X %*% B + w + tausq^.5 * rnorm(n)

set_missing <- rbinom(n, 1, 0.1)

simdata <- data.frame(coords,
                      y_full = y_full,
                      w_latent = w) %>%
  mutate(y_observed = ifelse(set_missing==1, NA, y_full))

# MCMC setup
mcmc_keep <- 500
mcmc_burn <- 100
```

```
mcmc_thin <- 2

y <- simdata$y_observed
ybar <- mean(y, na.rm=TRUE)

# training set
y_in <- (y-ybar)[!is.na(y)]
X_in <- X[!is.na(y),]
coords_in <- coords[!is.na(y),]

# suppose we dont want to have gridded knots
# i.e. we are fixing the MGP reference set at the observed locations
# (this may be inefficient in big data settings)
meshout <- spmeshed(y_in, X_in, coords_in,
                    axis_partition=c(4,4),
                    n_samples = mcmc_keep,
                    n_burn = mcmc_burn,
                    n_thin = mcmc_thin,
                    settings = list(forced_grid=FALSE, cache=FALSE),
                    prior=list(phi=c(1,15)),
                    verbose = 0,
                    n_threads = 1)

# test set
coords_out <- coords[is.na(y),]
X_out <- X[is.na(y),]

df_predict <- predict(meshout, newx=X_out, newcoords=coords_out)

y_posterior_predictive_mean <- df_predict$preds_out[,1,] %>%
  apply(1, mean) %>% add(ybar)
df_predicted <- df_predict$coords_out %>% cbind(y_posterior_predictive_mean)
```

---

rmeshedgp *Prior sampling from a Meshed Gaussian Process*

Description

Generates samples from a (univariate) MGP assuming a cubic directed acyclic graph and axis-parallel domain partitioning.

Usage

```
rmeshedgp(coords, theta,
  axis_partition = NULL, block_size = 100,
  n_threads=1, cache=TRUE, verbose=FALSE, debug=FALSE)
```

Arguments

- `coords` — matrix of spatial or spatiotemporal coordinates with $d = 2$ or $d = 3$ columns for spatial or spatiotemporal data, respectively.
- `theta` — vector with covariance parameters. If $d = 2$ and `theta` is a 2-dimensional vector then $\theta = (\phi, \sigma^2)$ where ϕ is the spatial decay and σ^2 is the spatial variance in the exponential covariance model. If $d = 2$ and `theta` is a 3-dimensional vector then $\theta = (\phi, \nu, \sigma^2)$ and a Matern model with smoothness ν is used instead. If $d = 3$, `theta` must be a 4-dimensional vector and $\theta = (a, \phi, b, \sigma^2)$ using Gneiting's non-separable spatiotemporal covariance detailed below.
- `axis_partition` — integer vector of length d with the number of intervals along which each axis should be partitioned. The domain will be partitioned into `prod(axis_partition)` blocks. This argument can be left blank when using `block_size`.
- `block_size` — integer specifying the (approximate) size of the blocks, i.e. how many spatial or spatiotemporal locations should be included in each block. Note: larger values correspond to an MGP that is closer to a full GP, but require more expensive computations.
- `n_threads` — integer number of OpenMP threads. This is ineffective if `meshed` was not compiled with OpenMP support.
- `cache` — bool: whether to use cache. Some computational speedup is associated to `cache=TRUE` if `coords` are a grid.
- `verbose` — bool: print some messages.
- `debug` — bool: print more messages.

Details

Gaussian processes (GPs) lack in scalability to big datasets due to the assumed unrestricted dependence across the spatial or spatiotemporal domain. *Meshed* GPs instead use a directed acyclic graph (DAG) with patterns, called *mesh*, to simplify the dependence structure across the domain. Each DAG node corresponds to a partition of the domain. MGPs can be interpreted as approximating the GP they originate from, or as standalone processes that can be sampled from. This function samples random MGPs and can thus be used to generate big spatial or spatiotemporal data. The only requirement to sample from a MGP compared to a standard GP is the specification of the domain partitioning strategy. Here, either `axis_partition` or `block_size` can be used; the default `block_size=100` can be used to quickly sample smooth surfaces at millions of locations.

Just like in a standard GP, one needs a covariance function or kernel which can be set as follows. For spatial data ($d = 2$), the length of `theta` determines which model is used (see above). Letting $h = \|s - s'\|$ where s and s' are locations in the spatial domain, the exponential covariance is defined as:

$$C(h) = \sigma^2 \exp\{-\phi h\},$$

whereas the Matern model is

$$C(h) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \phi^\nu h^\nu K_\nu(\phi h),$$

where K_ν is the modified Bessel function of the second kind of order ν. For spatiotemporal data ($d = 3$) the covariance function between locations (s, t) and (s', t') with distance $h = \|s - s'\|$ and time lag $u = \|t - t'\|$ is defined as

$$C(h, u) = \sigma^2/(au + 1)\exp\{-\phi h(au + 1)^{-b/2}\},$$

which is a special case of non-separable spacetime covariance as introduced by Gneiting (2002).

Value

data.frame with the (reordered) supplied coordinates in the first d columns, and the MGP sample in the last column, labeled w.

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>

References

Gneiting, T (2002) Nonseparable, Stationary Covariance Functions for Space-Time Data. *Journal of the American Statistical Association*. doi:10.1198/016214502760047113

Peruzzi, M., Banerjee, S., and Finley, A.O. (2020) Highly Scalable Bayesian Geostatistical Modeling via Meshed Gaussian Processes on Partitioned Domains. *Journal of the American Statistical Association*, in press. doi:10.1080/01621459.2020.1833889

Examples

```
library(ggplot2)
library(magrittr)
library(meshed)

# spatial domain (we choose a grid to make a nice image later)
# this generates a dataset of size 6400
xx <- seq(0, 1, length.out=80)
coords <- expand.grid(xx, xx) %>%
  as.matrix()

raster_plot <- function(df){
  ggplot(df, aes(Var1, Var2, fill=w)) +
    geom_raster() +
    scale_fill_viridis_c() +
    theme_minimal() }

# spatial data, exponential covariance
# phi=14, sigma^2=2
simdata <- rmeshedgp(coords, c(14, 2))
raster_plot(simdata)

# spatial data, matern covariance
# phi=14, nu=1, sigma^2=2
simdata <- rmeshedgp(coords, c(14, 1, 2))
raster_plot(simdata)

# spacetime data, gneiting's covariance
```

```
# 64000 locations
stcoords <- expand.grid(xx, xx, seq(0, 1, length.out=10))

# it should take less than a couple of seconds
simdata <- rmeshedgp(stcoords, c(1, 14, .5, 2))
# plot data at 7th time period
raster_plot(simdata %>% dplyr::filter(Var3==unique(Var3)[7]))
```

---

spmeshed *Posterior sampling for models based on MGPs*

Description

Fits Bayesian multivariate spatial or spatiotemporal regression models with latent MGPs via Markov chain Monte Carlo.

Usage

```
spmeshed(y, x, coords, k=NULL,
       family = "gaussian",
       axis_partition = NULL,
       block_size = 30,
       grid_size = NULL,
       grid_custom = NULL,
       n_samples = 1000,
       n_burnin = 100,
       n_thin = 1,
       n_threads = 4,
       verbose = 0,
       predict_everywhere = FALSE,
       settings = list(adapting=TRUE, forced_grid=NULL,
                       cache=NULL, ps=TRUE, saving=TRUE, low_mem=FALSE, hmc=4),
       prior = list(beta=NULL, tausq=NULL, sigmasq = NULL,
                       phi=NULL, a=NULL, nu = NULL,
                       toplim = NULL, btmlim = NULL, set_unif_bounds=NULL),
       starting = list(beta=NULL, tausq=NULL, theta=NULL, lambda=NULL, v=NULL,
                       a=NULL, nu = NULL,
                       mcmcsd=.05,
                       mcmc_startfrom=0),
       debug = list(sample_beta=TRUE, sample_tausq=TRUE,
                    sample_theta=TRUE, sample_w=TRUE, sample_lambda=TRUE,
                    verbose=FALSE, debug=FALSE),
       indpart=FALSE
)
```

Arguments

y — matrix of multivariate outcomes with n rows and q columns. Each row of y corresponds to a row of coords. NA values are accepted in any combination and will be predicted via MCMC.

x — matrix of covariates with n rows and p columns.

coords — matrix of coordinates with n rows and $d = 2$ or $d = 3$ columns for spatial or spacetime regression, respectively.

k — integer $k \leq q$, number of latent processes to use for the linear model of coregionalization. If unspecified, this is set to q=ncol(y).

family — a vector with length 1 or q whose elements corresponds to the data types of columns of y. Available choices are gaussian, poisson, binomial, beta for outcomes that are continuous, count, binary, or $(0, 1)$ proportions.

axis_partition — integer vector of size d: number of intervals each coordinate axis is split into

block_size — integer approximate size of the blocks after domain partitioning. Only used if axis_partition is not specified.

grid_size — integer vector of size d: number of 'knots' of the reference grid along each axis. This grid is then partitioned using either axis_partition or block_size. If unspecified, this is set so that the eventual grid size is close to n. This parameter is ignored if settings$forced_grid=FALSE in which case the data are assumed to be on a grid.

grid_custom — list with elements grid and axis_interval_partition. grid is a data.frame with the user supplied grid of knots. It is possible to include covariate values for the grid locations as additional columns, as long as their number matches ncol(x) - this is useful to make raster images of predictions. axis_interval_partition is the user supplied set of cuts for each coordinate axis (Note: these are the actual cutpoints along the axes, not the number of cuts). If left empty, axis_partition will be used to partition the custom grid. No checks are made on the validity of this grid. This parameter is ignored if settings$forced_grid=FALSE in which case the data are assumed to be on a grid.

n_samples — integer number of MCMC samples at which all the unknowns are stored (including the latent effects).

n_burnin — integer number of MCMC samples to discard at the beginning of the chain.

n_thin — integer thinning parameter for the MCMC chain. Only the chain of latent effects (w) is thinned to save memory in big data problems. Chains for other unknowns are not thinned and thus will be of length n_thin * n_samples.

n_threads — integer number of OpenMP threads. This is ineffective if meshed was not compiled with OpenMP support.

verbose — integer. If verbose<=20, then this is the number of times a message is displayed during MCMC. If verbose>20, then this is the number of MCMC iterations to wait until the next message update. If verbose=Inf, then a message will be printed at each MCMC iteration.

predict_everywhere — bool used if settings$forced_grid=T. Should predictions be made at the reference grid locations? If not, predictions will be made only at the supplied NA values of Y.

settings
: list: `settings$adapting` turns the adaptation of MCMC on/off, `settings$forced_grid` determines whether or not to use the data grid or a forced grid; if unspecified, the function will try to see what the data look like. Note: if `forced_grid=FALSE` and n is very large and $coords$ are irregularly spaced, then expect slowdowns in preprocessing and consider using `forced_grid=TRUE` instead. `settings$saving` will save model data if set to `TRUE`. `settings$low_mem` will only save `beta_mcmc`, `lambda_mcmc`, `v_mcmc`, `tausq_mcmc` (and not `w_mcmc` and `lp_mcmc`, which can be recovered from the others), thereby using less memory. All fitted predictions remain available in `yhat_mcmc` for convenience. `settings$ps` (default `TRUE`) determines whether to use the PS parametrization (Peruzzi et al 2021). `settings$hmc`, used if any outcome is not Gaussian, (1: MALA, 2: NUTS, 3: RM-MALA, 4: Simplified manifold preconditioning (default))

prior
: list: setup for priors of unknown parameters. `prior$phi` needs to be specified as the support of the Uniform prior for ϕ. There is currently limited functionality here and some inputs are currently ignored. Defaults are: a vague Gaussian for β, $\tau_i^2 \sim IG(2,1)$, $\theta_j \sim IG(2,2)$, all subject to change.

starting
: list: setup for starting values of unknown parameters. `starting$mcmcsd` is the initial standard deviation of proposals. `starting$mcmc_startfrom` is input to the adaptive MCMC and can be used to manually restart MCMC. There is currently limited functionality here and some parameters may be ignored.

debug
: list: setup for debugging things. Some parts of MCMC can be turned off here.

indpart
: bool defaults to `FALSE`. If `TRUE`, this computes an independent partition model.

Details

This function targets the following model:

$$y(s) = x(s)^\top \beta + \Lambda v(s) + \epsilon(s),$$

where $y(s)$ is a q-dimensional vector of outcomes at spatial location s, $x(s)$ is a p-dimensional vector of covariates with static coefficients β, Λ is a matrix of factor loadings of size (q,k), $v(s)$ is a k-dimensional vector which collects the realization of independent Gaussian processes $v_j \sim spmeshed(0, C_j)$ for $j = 1, \dots, k$ and where $C_j(s, s')$ is a correlation function. s is a coordinate in space ($d = 2$) or space plus time ($d = 3$). The Meshed GP implemented here associates an axis-parallel tessellation of the domain to a cubic directed acyclic graph (mesh).

Value

coordsdata
: data.frame including the original n coordinates plus the n_g knot coordinates if the model was run on a forced grid. The additional column `forced_grid` has value 1 if the corresponding coordinate is a knot in the forced grid. See examples.

savedata
: Available if `settings$saving==TRUE`. Needed for making predictions using `predict()` after MCMC. Note: `NA` values of the output are automatically and more efficiently predicted when running `spmeshed`.

yhat_mcmc
: list of length `n_samples` whose elements are matrices with $n + n_g$ rows and q columns. Each matrix in the list is a posterior predictive sample of the latent spatial process. $n_g = 0$ if the data grid is being used. Given the possibly large n, only the thinned chain is output for y.

v_mcmc — list of length n_samples whose elements are matrices with $n + n_g$ rows and k columns. Each matrix in the list is a posterior sample of the k latent spatial process. $n_g = 0$ if the data grid is being used. Given the possibly large n, only the thinned chain is output for v.

w_mcmc — list of length n_samples whose elements are matrices with $n + n_g$ rows and q columns. Each matrix in the list is a posterior sample of $w = \Lambda v$. $n_g = 0$ if the data grid is being used. Given the possibly large n, only the thinned chain is output for w.

lp_mcmc — list of length n_samples whose elements are matrices with $n + n_g$ rows and q columns. Each matrix in the list is a posterior sample of the linear predictor $X\beta + \Lambda v$. $n_g = 0$ if the data grid is being used. Given the possibly large n, only the thinned chain is output for w.

beta_mcmc — array of size (p, q, n_thin*n_samples) with the posterior sample for the static regression coefficients β. The jth column of each matrix (p rows and q columns) corresponds to the p linear effects on the jth outcome. The full chain minus burn-in is returned NOT thinned since p and q are relatively small.

tausq_mcmc — matrix of size (q, n_thin*n_samples). Each row corresponds to the full MCMC chain for the nugget τ^2_j of the jth outcome in the coregionalization/factor model. The full chain minus burn-in is returned NOT thinned since q is relatively small.

theta_mcmc — array of size (h, k, n_thin*n_samples) with the posterior sample for the correlation function parameters θ. h is 2 for spatial data (corresponding to the spatial decay of the exponential covariance ($\phi_i, i = 1, \dots, k$), and the variance $\sigma^2_i, i = 1, \dots, k$), 4 for spacetime data (corresponding to temporal decay, spatial decay, and separability – these are referred to as a_i, ϕ_i, and $\beta_i, i = 1, \dots, k$, in Gneiting (2002), see doi:10.1198/016214502760047113, plus the variance $\sigma^2, i = 1, \dots, k$). The full chain minus burn-in is returned NOT thinned since h and k are relatively small. If settings$ps=TRUE, the MCMC output for σ^2_i (last row of theta_mcmc) should be discarded and Λ used instead.

lambda_mcmc — array of size (q, k, n_thin*n_samples). Each matrix (of size (q, k)) is a posterior sample for Λ in the coregionalization/factor model. In univariate models, this is usually called σ. The full chain minus burn-in is returned NOT thinned since q and k are relatively small.

paramsd — Cholesky factorization of the proposal covariance for adaptive MCMC, after adaptation.

mcmc — Total number of MCMC iterations performed.

mcmc_time — Time in seconds taken for MCMC (not including preprocessing).

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>

References

Peruzzi, M., Banerjee, S., and Finley, A.O. (2022) Highly Scalable Bayesian Geostatistical Modeling via Meshed Gaussian Processes on Partitioned Domains. *Journal of the American Statistical Association*, 117(538):969-982. doi:10.1080/01621459.2020.1833889

Peruzzi, M., Banerjee, S., Dunson, D.B., and Finley, A.O. (2021) Grid-Parametrize-Split (GriPS) for Improved Scalable Inference in Spatial Big Data Analysis. https://arxiv.org/abs/2101.03579

Peruzzi, M. and Dunson, D.B. (2022) Spatial meshing for general Bayesian multivariate models. https://arxiv.org/abs/2201.10080

Examples

```
# toy example with tiny dataset and short MCMC
# on a univariate outcome
library(magrittr)
library(dplyr)
library(ggplot2)
library(meshed)

set.seed(2021)

SS <- 12
n <- SS^2 # total n. locations, including missing ones

coords <- expand.grid(xx <- seq(0,1,length.out=SS), xx) %>%
  as.matrix()

# generate data
sigmasq <- 2.3
phi <- 6
tausq <- .1
B <- c(-1,.5,1)

CC <- sigmasq * exp(-phi * as.matrix(dist(coords)))
LC <- t(chol(CC))
w <- LC %*% rnorm(n)
p <- length(B)
X <- rnorm(n * p) %>% matrix(ncol=p)
y_full <- X %*% B + w + tausq^.5 * rnorm(n)

set_missing <- rbinom(n, 1, 0.1)

simdata <- data.frame(coords,
                      y_full = y_full,
                      w_latent = w) %>%
  mutate(y_observed = ifelse(set_missing==1, NA, y_full))

# MCMC setup
mcmc_keep <- 500
mcmc_burn <- 100
mcmc_thin <- 2

y <- simdata$y_observed
ybar <- mean(y, na.rm=TRUE)

meshout <- spmeshed(y-ybar, X, coords,
```

```
                        axis_partition=c(4,4),
                        n_samples = mcmc_keep,
                        n_burn = mcmc_burn,
                        n_thin = mcmc_thin,
                        prior=list(phi=c(1,15)),
                        verbose = 0,
                        n_threads = 1)

# posterior means
best_post_mean <- meshout$beta_mcmc %>% apply(1:2, mean)

# process means
wmesh <- data.frame(w_mgp = meshout$w_mcmc %>% summary_list_mean())
# predictions
ymesh <- data.frame(y_mgp = meshout$yhat_mcmc %>% summary_list_mean())

outdf <-
  meshout$coordsdata %>%
  cbind(ymesh, wmesh)

# plot predictions
pred_plot <- outdf %>%
  ggplot(aes(Var1, Var2, color=y_mgp)) +
  geom_point() +
  scale_color_viridis_c()

# plot latent process
latent_plot <- outdf %>%
  ggplot(aes(Var1, Var2, color=w_mgp)) +
  geom_point() +
  scale_color_viridis_c()

# estimation of regression coefficients
plot(density(meshout$beta_mcmc[1,1,]))
abline(v=B[1], col="red")
```

`summary_list_mean` *Arithmetic mean of matrices in a list*

Description

For a list of matrices $\{X^{(1)}, \dots, X^{(L)}\}$, all of the same dimension, this function computes the matrix $\bar{X}$ with i, j entry $\bar{X}_{i,j} = \frac{1}{L} \sum_{l=1}^{L} X_{i,j}^{(l)}$. This function does not run any check on the dimensions and uses OpenMP if available.

Usage

```
summary_list_mean(x, n_threads=1)
```

Arguments

x	A list of matrices of the same dimension
n_threads	integer number of OpenMP threads. This is ineffective if meshed was not compiled with OpenMP support.

Value

The matrix of mean values.

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>

Examples

```
# make some data into a list
set.seed(2021)
L <- 200
x <- lapply(1:L, function(i) matrix(runif(300), ncol=3))
mean_done <- summary_list_mean(x)
```

---

summary_list_q *Quantiles of elements of matrices in a list*

Description

For a list of matrices $\{X^{(1)}, \dots, X^{(L)}\}$, all of the same dimension, this function computes the matrix $\hat{X}$ with i, j entry $\hat{X}_{i,j} =$ `quantile(`$\{X_{i,j}^{(l)}\}_{l=1}^{L}$`, q)`. This function does not run any check on the dimensions and uses OpenMP if available. This is only a convenience function that is supposed to speed up quantile computation for very large problems. The results may be slightly different from R's `quantile` which should be used for small problems.

Usage

```
summary_list_q(x, q, n_threads=1)
```

Arguments

x	A list of matrices of the same dimension.
q	A number between 0 and 1.
n_threads	integer number of OpenMP threads. This is ineffective if meshed was not compiled with OpenMP support.

Value

The matrix of quantiles.

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>

Examples

```
# make some data into a list
set.seed(2021)
L <- 200
x <- lapply(1:L, function(i) matrix(runif(300), ncol=3))
quant_done1 <- summary_list_q(x, .9)
```

Index