Package 'spamtree'

June 9, 2021

Type Package

Title Spatial Multivariate Trees

Version 0.2.1

Date 2021-06-08

Author Michele Peruzzi

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

Description Fits multivariate Bayesian spatial regression models for large datasets using Spatial Multivariate Trees (SpamTrees). The methods in this package are detailed in Peruzzi and Dunson (2020) <arXiv:2012.00943>.

License GPL-3

Imports Rcpp (>= 1.0.3), FNN, dplyr, magrittr, rlang, tibble

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown, abind, ggplot2

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-06-09 07:30:02 UTC

R topics documented:

`CrossCovarianceAG10` *Multivariate non-separable cross-covariance function on latent domain of variables.*

Description

This function implements the cross-covariance function used in Peruzzi and Dunson (2021), which is derived from eq. 7 in Apanasovich and Genton (2010).

Usage

```
CrossCovarianceAG10(coords1, mv1, coords2, mv2,
                    ai1, ai2, phi_i, thetamv, Dmat)
```

Arguments

`coords1`	matrix with spatial coordinates
`mv1`	integer vector with variable IDs. The length must match the number of rows in `coords1`
`coords2`	matrix with spatial coordinates
`mv2`	integer vector with variable IDs. The length must match the number of rows in `coords2`
`ai1`	q-dimensional vector
`ai2`	q-dimensional vector
`phi_i`	q-dimensional vector
`thetamv`	for bivariate data ($q = 2$), this is a scalar. For $q > 2$, this is a vector with elements α, β, ϕ.
`Dmat`	symmetric matrix of dimension (q, q) with zeroes on the diagonal and whose (i, j) element is $\delta_{i,j}$.

Details

Suppose we have q variables. For $h > 0$ and $\Delta > 0$ define:

$$C(h, \Delta) = \frac{\exp\{-\phi\|h\|/\exp\{\beta \log(1 + \alpha\Delta)/2\}\}}{\exp\{\beta \log(1 + \alpha\Delta)\}}$$

and for $j = 1, \ldots, q$, define $C_j(h) = \exp\{-\phi_j\|h\|\}$.

Then the cross-covariance between the ith margin of a q-variate process $w(\cdot)$ at spatial location s and the jth margin at location s' is built as follows. For $i = j$ as

$$Cov(w(s, \xi_i), w(s', \xi_j)) = \sigma_{i1}^2 C(h, 0) + \sigma_{i2}^2 C_i(h),$$

whereas if $i \neq j$ it is defined as

$$Cov(w(s, \xi_i), w(s', \xi_j)) = \sigma_{i1}\sigma_{i2} C(h, \delta_{ij}),$$

where ξ_i and ξ_j are the latent locations of margin i and j in the domain of variables and $\delta_{ij} = \|\xi_i - \xi_j\|$ is their distance in such domain.

Value

The cross-covariance matrix for all pairwise locations.

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>

References

Apanasovich, T. V. and Genton, M. G. (2010) Cross-covariance functions for multivariate random fields based on latent dimensions. *Biometrika*, 97:15-30. doi: 10.1093/biomet/asp078

Peruzzi, M. and Dunson, D. B. (2021) Spatial Multivariate Trees for Big Data Bayesian Regression. https://arxiv.org/abs/2012.00943

Examples

```
library(magrittr)
library(dplyr)
library(spamtree)

SS <- 10
xlocs <- seq(0.0, 1, length.out=SS)
coords <- expand.grid(xlocs, xlocs)
c1 <- coords %>% mutate(mv_id=1)
c2 <- coords %>% mutate(mv_id=2)

coords <- bind_rows(c1, c2)
coords_q <- coords %>% dplyr::select(-mv_id)
cx <- coords_q %>% as.matrix()
mv_id <- coords$mv_id

ai1 <- c(1, 1.5)
ai2 <- c(.1, .51)
phi_i <- c(1, 2)
thetamv <- 5

q <- 2
Dmat <- matrix(0, q, q)
Dmat[2,1] <- 1
Dmat[upper.tri(Dmat)] <- Dmat[lower.tri(Dmat)]

CC <- CrossCovarianceAG10(cx, mv_id, cx, mv_id, ai1, ai2, phi_i, thetamv, Dmat)
```

spamtree *Bayesian Spatial Multivariate Tree GP Regression*

Description

Bayesian linear multivariate spatial regression using SpamTrees.

Usage

```
spamtree(y, x, coords,
        mv_id = rep(1, length(y)),
        cell_size = 25,
        K = rep(2, ncol(coords)),
        start_level = 0,
        tree_depth = Inf,
        last_not_reference = TRUE,
        limited_tree = FALSE,
        cherrypick_same_margin = TRUE,
        cherrypick_group_locations = TRUE,
        mvbias = 0,
        mcmc = list(keep = 1000, burn = 0, thin = 1),
        num_threads = 4,
        verbose = FALSE,
        settings = list(adapting = TRUE, mcmcsd = 0.01,
          debug = FALSE, printall = FALSE),
        prior = list(set_unif_bounds = NULL,
          btmlim = NULL, toplim = NULL, vlim = NULL),
        starting = list(beta = NULL, tausq = NULL, theta = NULL, w = NULL),
        debug = list(sample_beta = TRUE, sample_tausq = TRUE,
          sample_theta = TRUE, sample_w = TRUE,
          sample_predicts = TRUE)
)
```

Arguments

y	vector of outcomes of size n. Correspondingly, `y[i]` is the observation of outcome `mv_id[i]` at location `coords[i,]` and with covariates `x[i,]`. This means that if the number of outcomes is $q > 1$ then these are all stacked in a vector, and their integer ID is stored in `mv_id`.
x	matrix of covariates with dimension (n, p).
coords	matrix of coordinates with dimension $(n, 2)$.
mv_id	integer vector of outcome IDs of size n with values in $\{1, \ldots, q\}$.
cell_size	integer number of knots for each node in the treed DAG. Defaults to 25. This is a target number and some nodes may include more or less locations. Here, knots can only be chosen among observed locations.

`K` integer vector of dimension 2 indicating the number of intervals for axis-parallel recursive partitioning. Each tree level will thus have `prod(K)` times as many partitions as the previous. Defaults to `c(2,2)`, leading to a recursive quadtree.

`start_level` integer indicating the root level. Example: `start_level=0` means there is 1 root node. `start_level=1` means there are `prod(K)` root nodes.

`tree_depth` integer indicating the number of branching steps in the tree. Defaults to `Inf` meaning that observed locations will be placed on tree nodes as much as possible.

`last_not_reference` bool indicating whether to treat the last level of the tree as a reference set. The default value `TRUE` is recommended when `tree_depth=Inf` or whenever only a very small number of observed locations remain at the last level.

`limited_tree` bool determining whether to use a recursive tree. If `TRUE`, each non-root node has 1 parent and `prod(K)` children. Otherwise, each node at level L (where root nodes have $L = 0$) has L parents.

`cherrypick_same_margin` bool used only for multivariate outcomes. This determines how to assign parents to leaf nodes. In a SpamTree, outcome j at a new spatial location is assigned the same parent of its nearest neighbor. If `cherrypick_same_margin=TRUE` then the nearest neighbor is searched within the subset of locations for which outcome j was observed. Otherwise, the nearest neighbor is searched within all locations at which any outcome was observed. If outcomes are aligned (all observed at the same locations) and `cherrypick_group_locations=TRUE`, then this setting has minimal or no effect.

`cherrypick_group_locations` bool used in multivariate settings to determine whether the allocation of knots to tree nodes should treat the q outcomes at location $s \in D$ as either (1) a q dimensional vector observed at 1 location, or (2) one observed outcome at each of q locations (i.e. same spatial location but different outcome index).

`mvbias` parameter used in settings of multivariate misalignment in which one or more outcomes are observed at a number of locations that is much smaller than others. `mvbias` can be used to disproportionately place the more sparsely observed outcomes near root nodes. This is justified by Prop. 1 in Peruzzi and Dunson (2021).

`mcmc` list for setting up MCMC. `mcmc$keep` is the number of MCMC samples to be saved, `mcmc$burn` is the number of iterations for burn-in, `mcmc$thin` is the thinning level for the chain. The total number of iterations that will be run is `burn + thin*keep`.

`num_threads` integer number of OpenMP threads to use within MCMC. Ineffective if source is compiled without OpenMP support.

`verbose` level of verbosity. All messages are suppressed if `verbose=FALSE` (default). It is useful to set `verbose=TRUE` for data of medium size or long MCMC chains.

`settings` list with additional settings. `settings$adapting` determines whether to use Robust Adaptive Metropolis algorithm of Vihola (2012). `settings$mcmcsd` is the initial standard deviation for the MCMC proposals before adaptation. `settings$debug` prints some debug messages. `settings$printall` determines whether to print to console at each iteration.

`prior`	setup for prior on θ, which currently only allows to specify the support of independent uniform distributions. See examples. (subject to change).
`starting`	list with starting values for all unknowns. Compatibility checks with `prior` are minimal and incompatible values may result in crashes.
`debug`	list with debug settings. Can be used to turn off parts of MCMC.

Details

This implements the following model (in stacked vector form):

$$y = X\beta + w + \epsilon,$$

where y is a n-dimensional vector of outcomes, X is a matrix of covariates of dimension (n, p) associated to coefficients β, w is a n-dimensional vector storing the realization of a spatial multivariate Gaussian tree $w(\cdot) \sim SpamTree_G(0, C_\theta)$ where G is a treed directed acyclic graph, and where $C_\theta(s, s')$ is a matrix-valued non-separable cross-covariance function on latent dimensions (see Peruzzi and Dunson (2021), equation 18, and `CrossCovarianceAG10`) where θ is a vector of unknown parameters. SpamTrees Gaussian processes are a scalable alternative to a spatial multivariate GP. Conditional independence across domain locations is assumed to be determined by the treed graph G, whose sparsity enables more efficient computations for the Gibbs sampler computed with this function. The graph architecture can be customized using inputs of the `spamtree` function. The example below computes SpamTrees on univariate data. A vignette exists with bivariate misaligned spatial data.

Value

`coords`	reordered spatial coordinates
`coordsinfo`	reordered spatial coordinates plus partitioning information.
`mv_id`	reordered outcome IDs.
`w_mcmc`	posterior sample of the spatial random effect. This is a list of length `mcmc$thin` whose elements are `n`-dimensional vectors of multivariate spatial random effects whose q margins are listed in `mv_id` as output here.
`yhat_mcmc`	posterior predictive sample. This is a list of length `mcmc$thin` whose elements are `n`-dimensional vectors of predictions whose q margins are listed in `mv_id` as output here.
`beta_mcmc`	array of size `c(p,mcmc$keep,q)` with posterior samples of the regression coefficients on each outcome. Example: `beta_mcmc[2,,1]` is the posterior sample for the second regressor on the first outcome.
`tausq_mcmc`	matrix with posterior samples of the q nuggets, one for each outcome.
`theta_mcmc`	matrix with posterior samples of the cross-covariance parameters. These include the latent distance between outcomes which may be poorly identifiable.
`mcmc_time`	elapsed clock time for MCMC.

Author(s)

Michele Peruzzi <michele.peruzzi@duke.edu>,
David B. Dunson <dunson@duke.edu>

References

Peruzzi, M. and Dunson, D. B. (2021) Spatial Multivariate Trees for Big Data Bayesian Regression. https://arxiv.org/abs/2012.00943

Vihola, M. (2012) Robust adaptive Metropolis algorithm with coerced acceptance rate. *Statistics and Computing*, 22:997-1008. doi: 10.1007/s1122201192695

Examples

```
# toy example with tiny dataset and short MCMC
# on a univariate outcome

library(magrittr)
library(dplyr)
library(ggplot2)
library(spamtree)

set.seed(2021)

SS <- 15
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 <- X %*% B + w + tausq^.5 * rnorm(n)

set_missing <- rbinom(n, 1, 0.1)

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

# MCMC setup
mcmc_keep <- 300
mcmc_burn <- 300
mcmc_thin <- 1

ybar <- mean(y, na.rm=TRUE)
```

```
# fit spamtree with defaults
spamtree_done <- spamtree(y - ybar, X, coords,
          mcmc = list(keep=mcmc_keep, burn=mcmc_burn, thin=mcmc_thin),
          num_threads = 1)

# predictions
y_out <- spamtree_done$yhat_mcmc %>%
  abind::abind(along=3) %>% `[`(,1,) %>%
  add(ybar) %>% apply(1, mean)
w_out <- spamtree_done$w_mcmc %>%
  abind::abind(along=3) %>% `[`(,1,) %>%
  apply(1, mean)

outdf <- spamtree_done$coordsinfo %>%
  cbind(data.frame(w_spamtree = w_out,
                   y_spamtree = y_out)) %>%
  left_join(simdata)

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

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

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

Index