Package ‘BSBT’

November 9, 2020

Title The Bayesian Spatial Bradley--Terry Model

Version 1.0.0

Description An implementation of the Bayesian Spatial Bradley--
Terry (BSBT) model. It can be used to investigate data sets where judges compared different spatial areas. It constructs a network to describe how the areas are connected, and then places a correlated prior distribution on the quality parameter for each area, based on the network. The package includes MCMC algorithms to estimate the quality parameters. The methodology is published in Seymour et. al. (2020) <arXiv:2010.14128>.

License GPL-3

Imports stats, MASS, igraph, lifecycle, utils

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Depends R (>= 2.10)

Suggests knitr, rmarkdown, testthat, sf, surveillance, spdep, RColorBrewer, deldir

VignetteBuilder knitr

RdMacros lifecycle

NeedsCompilation no

Author Rowland Seymour [aut, cre] (<https://orcid.org/0000-0002-8739-3921>), James Briant [aut]

Maintainer Rowland Seymour <rowland.seymour@nottingham.ac.uk>

Repository CRAN

Date/Publication 2020-11-09 20:10:05 UTC

R topics documented:

`BSBT` *BSBT: Bayesian Spatial Bradley–Terry*

Description

An implementation of the Bayesian Spatial Bradley–Terry (BSBT) model. It can be used to investigate data sets where judges compared different spatial areas. It constructs a network to describe how the areas are connected, and then places a correlated prior distribution on the quality parameter for each area, based on the network. The package includes MCMC algorithms to estimate the quality parameters.

Covariance Functions

The covariance functions can be used to construct the Multivariate Normal prior distribution. The prior distribution includes a constraint, where a linear combination of the parameters can be specified. There are two functions:

1. `constrained_adjacency_covariance_function` creates a covariance matrix using a network based metric, and
2. `constrained_covariance_function` creates a matrix using the Euclidean distance metric.

MCMC functions

The main MCMC function is `run_mcmc`, but in cases where the gender of the judges is known the function `run_gender_mcmc` can be used to analyse how the different genders behave.

comparisons_to_matrix *Construct Win Matrix from Comparisons*

Description

This function constructs a win matrix from a data frame of comparisons. It is needed for the MCMC functions.

Usage

```
comparisons_to_matrix(n.areas, comparisons)
```

Arguments

`n.areas`	The number of areas in the study.
`comparisons`	An N x 2 data frame, where N is the number of comparisons. Each row should correspond to a judgment. The first column is the winning area, the second column is the more losing area. The areas should be labeled from 1 to n.areas.

Value

A matrix where the i, j^th element is the number of times area i beat area j.

Examples

```
#Generate some sample comparisons
comparisons <- data.frame("winner" = c(1, 3, 2, 2), "loser" = c(3, 1, 1, 3))

#Create matrix from comparisons
win.matrix <- comparisons_to_matrix(3, comparisons)
```

constrained_adjacency_covariance_function *Construct a constrained covariance matrix from the adjacency matrix*

Description

This function constructs a covariance function from the graph's adjacency matrix. The covariance function may be squared exponential, rational quadratic or Matern. It includes a constraint, where a linear combination of the parameters can be fixed.

Usage

```
constrained_adjacency_covariance_function(
  adj.matrix,
  type,
  hyperparameters,
  linear.combination,
  linear.constraint = 0
)
```

Arguments

`adj.matrix` The graph adjacency matrix

`type` The type of covariance function used. One of "sqexp", "ratquad" or "matern". Note: only matern with nu = 5/2 is supported.

`hyperparameters` A vector containing the covariance function hyperparameters. For the squared exponential and matern, the vector should contain the variance and length scale, for the rational quadratic, the vector should contain the variance, length scale and scaling parameters

`linear.combination` A matrix which defines the linear combination of the parameter vector lambda = (lambda_1, ..., lambda_N)^T. The linear combination is a vector of coefficients such that linear.combination %*% lambda = linear.constraint.

`linear.constraint` The value the linear constraint takes. Defaults to 0.

Value

The mean vector and covariance matrix

See Also

For more information about covariance functions see https://www.cs.toronto.edu/~duvenaud/cookbook/ or http://www.gaussianprocess.org/gpml/chapters/RW4.pdf

Examples

```
#Construct covariance matrix of Dar es Salaam, Tanzania, using network metric
data(dar.adj.matrix, package = "BSBT") #load dar es salaam adjacency matrix
k <- constrained_adjacency_covariance_function(dar.adj.matrix, type = "sqexp",
       hyperparameters = c(1, 1), rep(1, dim(dar.adj.matrix)[1]), 0)
       #Covariance registetred by sum of subwards is 0 using rational quadratic function
```

constrained_covariance_function

Construct a constrained covariance matrix from the Euclidean coordinates of the objects

Description

This function constructs a covariance function from the Euclidean coordinates of the objects. The covariance function may be squared exponential, rational quadratic or Matern. It includes a constraint, where a linear combination of the parameters can be fixed.

Usage

```
constrained_covariance_function(
  coordinates,
  type,
  hyperparameters,
  linear.combination,
  linear.constraint = 0
)
```

Arguments

`coordinates` An Nx2 matrix containing the Euclidean coordinates of the nodes.

`type` The type of covariance function used. One of "sqexp", "ratquad" or "matern". Note: only matern with nu = 5/2 is supported.

`hyperparameters` A vector containing the covariance function hyperparameters. For the squared exponential and matern, the vector should contain the variance and length scale, for the rational quadratic, the vector should contain the variance, length scale and scaling parameters

`linear.combination` A matrix which defines the linear combination of the parameter vector lambda = (lambda_1, ..., lambda_N)^T. The linear combination is a vector of coefficients such that linear.combination %*% lambda = linear.constraint.

`linear.constraint` The value the linear constraint takes. Defaults to 0.

Value

The mean vector and covariance matrix

See Also

For more information about covariance functions see https://www.cs.toronto.edu/~duvenaud/cookbook/ or http://www.gaussianprocess.org/gpml/chapters/RW4.pdf

Examples

```
#Generate 10 points and create covariance matrix using Euclidean distance metric
coords <- data.frame("x" = c(0, 1, 2), "y" = c(0, 1, 2)) #generate coordinates
#create covariance matrix using Squared Exponential function and subject to the constraint
#the sum of the deprivation levels is 0.
k <- constrained_covariance_function(coords, "sqexp",
c(1, 5), rep(1, 3), linear.constraint = 0)
```

---

dar.adj.matrix *Adjacency matrix for the subwards in Dar es Salaam, Tanzania*

Description

Adjacency matrix for the subwards in Dar es Salaam, Tanzania

Usage

```
dar.adj.matrix
```

Format

A 452x452 matrix, where a_ij = 1 if subwards i and j are neighbours and 0 otherwise. The adjacency matrix is based on areas which share administrative borders. Two additional edges over the Kurasini creek to represent a road and ferry crossing have been added.

---

dar.comparisons *Comparative Judgment on Deprivation in Dar es Salaam, Tanzania*

Description

A comparative judgment data set on deprivation in subwards in Dar es Salaam, Tanzania. Citizens were shown pairs of subwards at random and asked which was more deprived. If they said they were equal, one of the pair was chosen at random to be more deprived. The data was collected in August 2018. The gender of each judge is also included.

Usage

```
dar.comparisons
```

Format

A csv file containing 75078 rows and 3 columns. Each row corresponds to a judgement made by a single judge. Columns 2 and 3 shows which of the pair of subwards was judged to be poorest and richest, and column 3 shows the gender of the judge.

Source

This data set was collected by Madeleine Ellis, James Goulding, Bertrand Perrat, Gavin Smith and Gregor Engelmann.We gratefully acknowledge the Rights Lab at the University of Nottingham for supporting funding for the comprehensive ground truth survey. We also acknowledge HumanitarianStreet Mapping Team (HOT) for providing a team of experts in data collection to facilitate the surveys. This work was also supported by the EPSRC Horizon Centre for Doctoral Training - My Life in Data (EP/L015463/1) and EPSRC grant Neodemographics (EP/L021080/1).

`dar.shapefiles` *Shape files for the subwards in Dar es Salaam, Tanzania*

Description

Polygons for the 452 subwards in Dar es Salaam, Tanzania

Usage

```
dar.shapefiles
```

Format

A .shp object

`female.mean.deprivation` *The mean level of deprivation for subwards in Dar es Salaam as perceived by women*

Description

This data is used in the vignette

Usage

```
female.mean.deprivation
```

Format

An vector of 452 elements, one for each subward

`loglike_function` *Compute the loglikelihood function*

Description

This function computes the BSBT model loglikelihood function. It requires the deprivation levels and the win matrix.

Usage

```
loglike_function(x, win.matrix)
```

Arguments

`x`	The level of deprivation of the areas on an exponential scale
`win.matrix`	A matrix, where w_ij give the number of times area i beat j

Value

The value of of the loglikelihood function

Examples

```
win.matrix <- matrix(c(0, 3, 2, 1, 0, 1, 1, 3, 0), 3, 3) #construct win matrix
lambda     <- c(3, 1, 2)

l <- loglike_function(lambda, win.matrix)
```

`male.mean.deprivation` *The mean level of deprivation for subwards in Dar es Salaam as perceived by men*

Description

This data is used in the vignette

Usage

```
male.mean.deprivation
```

Format

An vector of 452 elements, one for each subward `mean.deprivation` *The Mean Level of Deprivation for Subwards in Dar es Salaam*

Description

This data is used in the vignette

Usage

```
mean.deprivation
```

Format

An vector

`run_asymmetric_mcmc` *Run the BSBT MCMC algorithm with n types of individuals and asymmetric variance*

Description

This function runs the MCMC algorithm with n types of individuals, for example male and female. The types must share the same covariance matrix and the win matrices are entered as a list. The first item in the list acts as the baseline group. This model has an asymmetric variance structure, as the variance of the baseline is always smaller. For a model with thee types, f, g and h, the structure is as follows. The baseline is f, or the second type, g = f + d_1, and the third type, h = f + d_2. Here d_1 and d_2 are the discrepancy between each type and the baseline.

Usage

```
run_asymmetric_mcmc(
  n.iter,
  delta,
  covariance.matrix,
  win.matrices,
  estimates.initial,
  omega = 0.1,
  chi = 0.1
)
```

Arguments

`n.iter` The number of iterations to be run

`delta` The underrlaxed tuning parameter must be in (0, 1)

`covariance.matrix` The output from the covariance matrix function, which contains the decomposed and inverted covariance matrix. The variance hyperparameter must be set to 1.

`win.matrices` A list of n matrices where the ith matrix is the win matrix corresponding to only the ith level

`estimates.initial` A list of vectors where the ith vector is the initial estimate for the ith level effect

`omega` The value of the inverse gamma shape parameter

`chi` The value of the inverse gamma scale parameter

Value

A list of MCMC output

- estimates - A list of matrices. Each matrix containing the iteration of the ith level
- alpha.sq - A matrix containing the iterations of alpha^2
- acceptance.rate - The acceptance rate for f and g
- time.taken - Time taken to run the MCMC algorithm in seconds

Examples

```
n.iter <- 10
delta <- 0.1
covariance.matrix <- list()
covariance.matrix$mean <- c(0, 0, 0)
covariance.matrix$decomp <- diag(3)
covariance.matrix$inv    <- diag(3)
men.comparisons <- data.frame("winner" = c(1, 3, 2, 2), "loser" = c(3, 1, 1, 3))
women.comparisons <- data.frame("winner" = c(1, 2, 1, 2), "loser" = c(3, 1, 3, 3))
men.win.matrix <- comparisons_to_matrix(3, men.comparisons)
women.win.matrix <- comparisons_to_matrix(3, women.comparisons)
f.initial <- c(0, 0, 0)
g.initial <- c(0, 0, 0)

win.matrices <- list(men.win.matrix, women.win.matrix)
estimates.initial <- list(f.initial, g.initial)

mcmc.output<- run_asymmetric_mcmc(n.iter, delta, covariance.matrix, win.matrices, estimates.initial)
```

run_gender_mcmc — *Run the BSBT with Gender Effect MCMC algorithm*

Description

This function runs the BSBT MCMC algorithm where the male and female judges can be separated. It generates samples for the grand mean of the male and female perceptions for the derivation in each area and the difference between them. It is similar to `run_mcmc`. This function requires the data to be separate into two parts, one for each gender. There should be a win matrix for the male judges, and a win matrix for the female judges. Similarly, initial estimates for the grand mean and difference parameters need to be included seperately.

Usage

```
run_gender_mcmc(
  n.iter,
  delta,
  covariance.matrix,
  male.win.matrix,
  female.win.matrix,
  f.initial,
  g.initial,
  omega = 0.1,
  chi = 0.1,
  thinning = 1
)
```

Arguments

`n.iter` The number of iterations to be run

`delta` The underrlaxed tuning parameter. Must be in (0, 1)

`covariance.matrix` The output from the covariance matrix function, which contains the decomposed and inverted covariance matrix. The variance hyperparameter must be set to 1.

`male.win.matrix` A matrix, where w_ij give the number of times area i beat j when judged by men

`female.win.matrix` A matrix, where w_ij give the number of times area i beat j when judged by women

`f.initial` A vector of the initial estimate for f, the grand mean of men and women's perceptions

`g.initial` A vector of the initial estimate for g, the difference between men and women's perceptions

`omega` The value of the inverse gamma shape parameter

`chi` The value of the inverse gamma scale parameter `thinning` Setting thinning to i will store every i^th iteration. This may be required for very long runs.

Value

A list of MCMC output

- f.matrix - A matrix containing the each iteration of f
- g.matrix - A matrix containing the each iteration of g
- alpha.sq - A matrix containing the iterations of alpha^2
- acceptance.rate - The acceptance rate for f and g
- time.taken - Time taken to run the MCMC algorithm in seconds

Examples

```
n.iter <- 10
delta <- 0.1
covariance.matrix <- list()
covariance.matrix$mean <- c(0, 0, 0)
covariance.matrix$decomp <- diag(3)
covariance.matrix$inv    <- diag(3)
men.comparisons <- data.frame("winner" = c(1, 3, 2, 2), "loser" = c(3, 1, 1, 3))
women.comparisons <- data.frame("winner" = c(1, 2, 1, 2), "loser" = c(3, 1, 3, 3))
men.win.matrix <- comparisons_to_matrix(3, men.comparisons) #win matrix for the male judges
women.win.matrix <- comparisons_to_matrix(3, women.comparisons) #win matrix for the female judges
f.initial <- c(0, 0, 0) #initial estimate for grand mean
g.initial <- c(0, 0, 0) #initial estimate for differences

mcmc.output <- run_gender_mcmc(n.iter, delta, covariance.matrix, men.win.matrix,
    women.win.matrix, f.initial, g.initial)
```

`run_mcmc` *Run the BSBT MCMC algorithm*

Description

This function runs the BSBT MCMC algorithm to estimate the deprivation parameters. In this version, the judges are assumed to act homogeneously. This algorithm estimates the deprivation in each area and the prior distribution variance parameter. For data with two types of judges, see `run_gender_mcmc`.

Usage

```
run_mcmc(
  n.iter,
  delta,
  covariance.matrix,
  win.matrix,
  f.initial,
  alpha = FALSE,
  omega = 0.1,
  chi = 0.1
)
```

Arguments

`n.iter`	The number of iterations to be run
`delta`	The underrlaxed tuning parameter must be in (0, 1)
`covariance.matrix`	The output from the covariance matrix function, which contains the decomposed and inverted covariance matrix.
`win.matrix`	A matrix, where w_ij give the number of times area i beat j
`f.initial`	A vector of the initial estimate for f
`alpha`	A boolean if inference for alpha should be carried out. If this is TRUE, the covariance matrix
`omega`	The value of the inverse gamma shape parameter
`chi`	The value of the inverse gamma scale parameter

Value

A list of MCMC output

- f.matrix - A matrix containing the each iteration of f
- alpha.sq - A vector containing the iterations of alpha^2
- acceptance.rate - The acceptance rate for f
- time.taken - Time taken to run the MCMC algorithm in seconds

Examples

```
n.iter <- 10
delta <- 0.1
covariance.matrix <- list()
covariance.matrix$mean <- c(0, 0, 0)
covariance.matrix$decomp <- diag(3)
covariance.matrix$inv    <- diag(3)
comparisons <- data.frame("winner" = c(1, 3, 2, 2), "loser" = c(3, 1, 1, 3))
win.matrix <- comparisons_to_matrix(3, comparisons) #construct covariance matrix
f.initial <- c(0, 0, 0) #initial estimates for lamabda_1, lambda_2, lambda_3
```

```
mcmc.output <- run_mcmc(n.iter, delta, covariance.matrix, win.matrix, f.initial)
```

---

run_mcmc_with_ordering
Run the BSBT MCMC algorithm with ordering constraints

Description

This function runs the BSBT MCMC algorithm with ordering constraints. This allows the sign of lambda_i - lambda_j to be specified. The confidence parameters specify the confidence in this constraint. As this parameter approaches 0, all proposals that do not meet this constraint will be rejected. As this parameter approaches infinity, all proposals are accepted, regardless of the constraint. Only small numbers of ordering constraints should be included, as they can affect the mixing of the markov chain.

Usage

```
run_mcmc_with_ordering(
  n.iter,
  delta,
  covariance.matrix,
  win.matrix,
  f.initial,
  S,
  alpha = FALSE,
  omega = 0.1,
  chi = 0.1
)
```

Arguments

`n.iter`	The number of iterations to be run
`delta`	The underrlaxed tuning parameter must be in (0, 1)
`covariance.matrix`	The output from the covariance matrix function, which contains the decomposed and inverted covariance matrix.
`win.matrix`	A matrix, where w_ij give the number of times area i beat j
`f.initial`	A vector of the initial estimate for f
`S`	A list of ordering constraints. There are four elements in each set, the label of the two areas, the value of the constraints, and the confidence parameter; S = (i, j, ±1, nu).
`alpha`	A boolean if inference for alpha should be carried out. If this is TRUE, the covariance matrix

`omega`	The value of the inverse gamma shape parameter
`chi`	The value of the inverse gamma scale parameter

Value

A list of MCMC output

- f.matrix - A matrix containing the each iteration of f
- alpha.sq - A vector containing the iterations of alpha^2
- acceptance.rate - The acceptance rate for f
- time.taken - Time taken to run the MCMC algorithm in seconds

Examples

```
n.iter <- 10
delta <- 0.1
covariance.matrix <- list()
covariance.matrix$mean <- c(0, 0, 0)
covariance.matrix$decomp <- diag(3)
covariance.matrix$inv    <- diag(3)
comparisons <- data.frame("winner" = c(1, 3, 2, 2), "loser" = c(3, 1, 1, 3))
win.matrix <- comparisons_to_matrix(3, comparisons)
f.initial <- c(0, 0, 0)
S <- list()
S[[1]] <- c(1, 3, -1, 3) #Specify that lambda_1 - lambda_3 < 0,
#and the confidence parameter has value 3.
S[[2]] <- c(1, 2, -1, 3) #Specify that lambda_1 - lambda_2 < 0,
#and the confidence parameter has value 3.
mcmc.output <- run_mcmc_with_ordering(n.iter, delta, covariance.matrix, win.matrix, f.initial, S)
```

`simulate_comparisons` *Simulate contests from the Bradley–Terry Model*

Description

This function simulates pair-wise contests according to the Bradley–Terry model. It require the true quality of the areas and the number of comparisons to be carried out. It can also include some judge noise or error. When including noise, each time a judge carries out a comparisons, we assume they use the true quality with some zero-mean normal noise added. The standard deviation must be specified.

Usage

```
simulate_comparisons(n.contests, true.quality, sigma.obs)
```

Arguments

`n.contests`	The number of contests to be carried out
`true.quality`	A vector with the level of deprivation in each area on the log scale.
`sigma.obs`	Standard deviation for the noise to be added to the level of deprivation in each area. If 0, no noise is used.

Value

A list containing a data.frame with each pair-wise contest, the outcome (a 1 for a win, a 0 for a loss), and a win matrix where the i,j^th element is the number of times i beat j

Examples

```
example.deprivation <- -2:2 #True level of deprivation in each area
example.comparisons <- simulate_comparisons(10, example.deprivation, 0)
#generate comparisons with judge noise.
example.comparisons <- simulate_comparisons(10, example.deprivation, 0.1)
```

Index