Package: netcmc (via r-universe)

August 29, 2024

Type Package

Title Spatio-Network Generalised Linear Mixed Models for Areal Unit and Network Data

Version 1.0.2

Date 2022-11-07

Author George Gerogiannis, Mark Tranmer, Duncan Lee

Maintainer George Gerogiannis <`g.gerogiannis.1@research.gla.ac.uk`>

Description Implements a class of univariate and multivariate spatio-network generalised linear mixed models for areal unit and network data, with inference in a Bayesian setting using Markov chain Monte Carlo (MCMC) simulation. The response variable can be binomial, Gaussian, or Poisson. Spatial autocorrelation is modelled by a set of random effects that are assigned a conditional autoregressive (CAR) prior distribution following the Leroux model (Leroux et al. (2000) <doi:10.1007/978-1-4612-1284-3_4>). Network structures are modelled by a set of random effects that reflect a multiple membership structure (Browne et al. (2001) <doi:10.1177/1471082X0100100202>).

License GPL (>= 2)

Depends R (>= 4.0.0), MCMCpack

Imports Rcpp (>= 1.0.4), coda, ggplot2, mvtnorm, MASS

LinkingTo Rcpp, RcppArmadillo, RcppProgress

LazyLoad yes

Suggests testthat, igraph, magic

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-11-08 22:30:15 UTC

Contents

netcmc-package . . . 2
getAdjacencyMatrix . . . 3
getMembershipMatrix . . . 5
getTotalAltersByStatus . . . 6
multiNet . . . 7
multiNetLeroux . . . 10
multiNetRand . . . 14
plot.netcmc . . . 17
print.netcmc . . . 18
summary.netcmc . . . 18
uni . . . 19
uniNet . . . 21
uniNetLeroux . . . 25
uniNetRand . . . 30

Index **36**

`netcmc-package` *An R Package for Bayesian Social Network Modelling*

Description

Implements a class of univariate and multivariate spatio-network generalised linear mixed models, with inference in a Bayesian setting using Markov chain Monte Carlo (MCMC) simulation. The response variable can be binomial, Gaussian, and Poisson.

Details

Package:	netcmc
Type:	Package
Version:	1.0
Date:	2022-01-24
License:	GPL (>= 2)

Author(s)

George Gerogiannis <g.gerogiannis.1@research.gla.ac.uk>

Examples

```
## See the examples in the function specific help files.
```

`getAdjacencyMatrix` *A function that extracts valuable properties from a raw social network.*

Description

This function transforms a network, which is a data.frame type in a specified format, in to a resultant n by n adjacency matrix, where a_{ij} = 0 if vertex i and j $(i \neq j)$ are not adjacent i.e. vertex i and j are not the head/tail of an edge e and a_{ij} = 1 if vertex i and j $(i \neq j)$ are adjacent i.e. vertex i and j are the head/tail of an edge e. a_{ij} = 0 when $i = j$.

Usage

```
getAdjacencyMatrix(rawNetwork)
```

Arguments

`rawNetwork`	The data.frame which encodes information about the network. The dimensions of the matrix are n by $(l+1)$.The data.frame contains one column corresponding to the labels for each of the n vertices in the network, the column name for this should be 'labels'. The other l columns corresponds to the corresponds to the vertices which are adjacent to each of the n vertices in the network. It is important to note that the label of a vertex should not be 0. The nth vertex can be adjacent to a maximum of l other vertices.

Value

`adjacencyMatrix`	The resultant adjaceny matrix for the rawNetwork data.frame.
`nonnominators`	The individuals in the social network who are nominees of at least one other individual but were not in the set of individuals who did the nominating.
`vertexNoOutdegrees`	The individuals in the social network that have an outdegree of 0.
`vertexNoIndegrees`	The individuals in the social network that have an indegree of 0.
`vertexIsolates`	The individuals in the social network that have an outdegree and indegree of 0.

Author(s)

George Gerogiannis

Examples

```
rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[1] = "labels"
rawNetwork[, 1] = c("A", "B", "C", "D")
rawNetwork[, 2] = c(0, "C", "D", 0)
```

```
rawNetwork[, 3] = c("B", 0, "A", "C")
getAdjacencyMatrix(rawNetwork)

rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[2] = "labels"
rawNetwork[, 1] = c(NA, "Charlie", "David", 0)
rawNetwork[, 2] = c("Alistar", "Bob", "Charlie", "David")
rawNetwork[, 3] = c("Bob", NA, "Alistar", "Charlie")
getAdjacencyMatrix(rawNetwork)

rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[1] = "labels"
rawNetwork[, 1] = c(245, 344, 234, 104)
rawNetwork[, 2] = c(NA, 234, 104, NA)
rawNetwork[, 3] = c(344, 0, 245, 234)
getAdjacencyMatrix(rawNetwork)

rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[1] = "labels"
rawNetwork[, 1] = c(245, 344, 234, 104)
rawNetwork[, 2] = c(32, 234, 104, 0)
rawNetwork[, 3] = c(344, 20, 245, 234)
getAdjacencyMatrix(rawNetwork)

rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[1] = "labels"
rawNetwork[, 1] = c("Alistar", "Bob", "Charlie", "David")
rawNetwork[, 2] = c(NA, "Charlie", "David", 0)
rawNetwork[, 3] = c("Bob", "Blaine", "Alistar", "Charlie")
getAdjacencyMatrix(rawNetwork)

rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[1] = "labels"
rawNetwork[, 1] = c("Alistar", "Bob", "Charlie", "David")
rawNetwork[, 2] = c(0, "Charlie", 0, 0)
rawNetwork[, 3] = c("Bob", "Blaine", "Alistar", 0)
getAdjacencyMatrix(rawNetwork)

rawNetwork = matrix(NA, 4, 3)
rawNetwork = as.data.frame(rawNetwork)
colnames(rawNetwork)[1] = "labels"
rawNetwork[, 1] = c(245, 344, 234, 104)
rawNetwork[, 2] = c(32, 0, 104, 0)
rawNetwork[, 3] = c(34, 0, 245, 234)
getAdjacencyMatrix(rawNetwork)
```

`getMembershipMatrix` *A function that generates a data.frame that is the membership matrix of the network.*

Description

A function that generates a data.frame that is the membership matrix of the network given individual IDs and the alters that they have nominated.

Usage

```
getMembershipMatrix(individualID, alters)
```

Arguments

`individualID`	A data.frame which stores the IDs of the individuals that nominate alters.
`alters`	A data.frame which stores the alters of a given individual.

Value

`membershipMatrix`	The resultant data.frame.

Author(s)

George Gerogiannis

Examples

```
individualID = data.frame(c(1, 2, 3))
alters = data.frame(c(5, 3, 2), c(5, 6, 1))
getMembershipMatrix(individualID, alters)

individualID = data.frame(c(1, 2, 3))
alters = data.frame(c(NA, 3, 2), c(NA, NA, 1))
getMembershipMatrix(individualID, alters)

individualID = data.frame(c(1, 2, 3))
alters = data.frame(c(NA, 3, NA), c(NA, NA, 1))
getMembershipMatrix(individualID, alters)

individualID = data.frame(c(1, 2, 3))
alters = data.frame(c(NA, 3, NA), c(6, NA, 1))
getMembershipMatrix(individualID, alters)
```

`getTotalAltersByStatus` *A function that generates a data.frame that stores the number of alters with a given level of a factor an individual has.*

Description

This is a function that can be used to generates a data.frame that stores the number of alters with a given level of a factor an individual has.

Usage

```
getTotalAltersByStatus(individualID, status, alters)
```

Arguments

`individualID`	A data.frame which stores the IDs of the individuals that nominate alters.
`status`	A data.frame which stores the levels of a variable.
`alters`	A data.frame which stores the alters of a given individual.

Value

`totalAltersByStatus`	The resultant data.frame.

Author(s)

George Gerogiannis

Examples

```
individualID = data.frame(c(1, 2, 3, 4))
status = data.frame(c(10, 20, 30, 20))
alters = data.frame(c(4, 3, 2, 1), c(3, 4, 1, 2), c(2, 1, 4, 3))
totalAltersByStatus = getTotalAltersByStatus(individualID, status, alters)

individualID = data.frame(c(1, 2, 3, 4))
status = data.frame(c("RegularSmoke", "Nonsmoker", "CasualSmoker", "Nonsmoker"))
alters = data.frame(c(4, 3, 2, 1), c(3, 4, 1, 2), c(5, 1, 5, 3))
totalAltersByStatus = getTotalAltersByStatus(individualID, status, alters)

individualID = data.frame(c(1, 2, 3, 4))
status = data.frame(c(NA, "Nonsmoker", "CasualSmoker", "Nonsmoker"))
alters = data.frame(c(4, 3, 2, 1), c(3, 4, 1, 2), c(5, 1, 5, 3))
totalAltersByStatus = getTotalAltersByStatus(individualID, status, alters)

individualID = data.frame(c(10, 20))
status = data.frame(c(NA, "Nonsmoker"))
alters = data.frame(c(NA, 10), c(20, NA))
```

```
totalAltersByStatus = getTotalAltersByStatus(individualID, status, alters)

individualID = data.frame(c(NA, 20))
status = data.frame(c("Smoker", "Nonsmoker"))
alters = data.frame(c(NA, 10), c(20, NA))
totalAltersByStatus = getTotalAltersByStatus(individualID, status, alters)
```

`multiNet` *A function that generates samples for a multivariate fixed effects and network model.*

Description

This function that generates samples for a multivariate fixed effects and network model, which is given by

$$
\begin{gathered}
Y_{i_sr}|\mu_{i_sr} \sim f(y_{i_sr}|\mu_{i_sr}, \sigma^2_{er}) \quad i=1,\dots,N_s,\ s=1,\dots,S,\ r=1,\dots,R, \\
g(\mu_{i_sr}) = \boldsymbol{x}_{i_s}^\top \boldsymbol{\beta}_r + \sum_{j \in \text{net}(i_s)} w_{i_sj} u_{jr} + w^*_{i_s} u^*_r, \\
\boldsymbol{\beta}_r \sim \mathrm{N}(\mathbf{0}, \alpha \boldsymbol{I}) \\
\boldsymbol{u}_j = (u_{1j}, \dots, u_{Rj}) \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{u}}), \\
\boldsymbol{u}^* = (u^*_1, \dots, u^*_R) \sim \mathrm{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{u}}), \\
\boldsymbol{\Sigma}_{\boldsymbol{u}} \sim \text{Inverse-Wishart}(\xi_{\boldsymbol{u}}, \boldsymbol{\Omega}_{\boldsymbol{u}}), \\
\sigma^2_{er} \sim \text{Inverse-Gamma}(\alpha_3, \xi_3).
\end{gathered}
$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p \times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p \times 1$ vector of fixed effect parameters relating to the rth response are denoted by $\boldsymbol{\beta}_r$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha \boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ^2_{er}, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

The $R \times 1$ vector of random effects for the jth alter is denoted by $\boldsymbol{u}_j = (u_{j1}, \dots, u_{jR})_{R\times 1}$, while the $R \times 1$ vector of isolation effects for all R outcomes is denoted by $\boldsymbol{u}^* = (u^*_1, \dots, u^*_R)$, and both are assigned multivariate Gaussian prior distributions. The unstructured covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ captures the covariance between the R outcomes at the network level, and a conjugate Inverse-Wishart prior is specified for this covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}$. The corresponding hyperparamaterers ($\xi_{\boldsymbol{u}}$, $\boldsymbol{\Omega}_{\boldsymbol{u}}$) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$
\begin{gathered}
\text{Binomial: } Y_{i_sr} \sim \text{Binomial}(n_{i_sr}, \theta_{i_sr}) \text{ and } g(\mu_{i_sr}) = \ln(\theta_{i_sr}/(1-\theta_{i_sr})), \\
\text{Gaussian: } Y_{i_sr} \sim \mathrm{N}(\mu_{i_sr}, \sigma^2_{er}) \text{ and } g(\mu_{i_sr}) = \mu_{i_sr}, \\
\text{Poisson: } Y_{i_sr} \sim \text{Poisson}(\mu_{i_sr}) \text{ and } g(\mu_{i_sr}) = \ln(\mu_{i_sr}).
\end{gathered}
$$

Usage

```
multiNet(formula, data, trials, family, W, numberOfSamples = 10, burnin = 0,
thin = 1, seed = 1, trueBeta = NULL, trueURandomEffects = NULL,
trueVarianceCovarianceU = NULL, trueSigmaSquaredE = NULL,
covarianceBetaPrior = 10^5, xi, omega, a3 = 0.001, b3 = 0.001,
centerURandomEffects = TRUE)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials $n_{i_s r}$. Only used if `family`=“binomial".
`family`	The data likelihood model that must be “gaussian", “poisson" or “binomial".
`W`	A matrix $\boldsymbol{W}$ that encodes the social network structure and whose rows sum to 1.
`numberOfSamples`	The number of samples to generate pre-thin.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true value of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_R$.
`trueURandomEffects`	If available, the true values of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J, \boldsymbol{u}^*$.
`trueVarianceCovarianceU`	If available, the true value of $\boldsymbol{\Sigma}_{\boldsymbol{u}}$.
`trueSigmaSquaredE`	If available, the true value of σ^2_{e1}, …, σ^2_{eR}. Only used if `family`=“gaussian".
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha\boldsymbol{I}$.
`xi`	The degrees of freedom parameter for the Inverse-Wishart distribution relating to the network random effects $\xi_{\boldsymbol{u}}$.
`omega`	The scale parameter for the Inverse-Wishart distribution relating to the network random effects $\boldsymbol{\Omega}_{\boldsymbol{u}}$.
`a3`	The shape parameter for the Inverse-Gamma distribution relating to the error terms α_3. Only used if `family`=“gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution relating to the error terms ξ_3. Only used if `family`=“gaussian".
`centerURandomEffects`	A choice to center the network random effects after each iteration of the MCMC sampler.

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`W`	The network matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_R$ parameters in the model.
`varianceCovarianceUSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ in the model.
`uRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of network random effects $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J, \boldsymbol{u}^*$ in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of σ^2_{e1}, …, σ^2_{eR} in the model. Only used if `family`=“gaussian".
`acceptanceRates`	The acceptance rates of parameters in the model from the MCMC sampling scheme .
`uRandomEffectsAcceptanceRate`	The acceptance rates of network random effects in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`DBar`	DBar for the model.
`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

`multiNetLeroux` *A function that generates samples for a multivariate fixed effects, spatial, and network model.*

Description

This function that generates samples for a multivariate fixed effects, spatial, and network model, which is given by

$$
\begin{aligned}
Y_{i_s r} | \mu_{i_s r} &\sim f(y_{i_s r} | \mu_{i_s r}, \sigma^2_{er}) \quad i = 1, \ldots, N_s,\ s = 1, \ldots, S,\ r = 1, \ldots, R, \\
g(\mu_{i_s r}) &= \boldsymbol{x}^\top_{i_s} \boldsymbol{\beta}_r + \phi_{sr} + \sum_{j \in \text{net}(i_s)} w_{i_s j} u_{jr} + w^*_{i_s} u^*_r, \\
\boldsymbol{\beta}_r &\sim \text{N}(\mathbf{0}, \alpha \boldsymbol{I}) \\
\boldsymbol{\phi}_r = (\phi_{1r}, \ldots, \phi_{Sr}) &\sim \text{N}(\mathbf{0}, \tau^2_r(\rho_r(\text{diag}(\boldsymbol{A}\mathbf{1}) - \boldsymbol{A}) + (1 - \rho_r)\boldsymbol{I})^{-1}), \\
\boldsymbol{u}_j = (u_{1j}, \ldots, u_{Rj}) &\sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma_u}), \\
\boldsymbol{u}^* = (u^*_1, \ldots, u^*_R) &\sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma_u}), \\
\tau^2_r &\sim \text{Inverse-Gamma}(a_1, b_1), \\
\rho_r &\sim \text{Uniform}(0, 1), \\
\boldsymbol{\Sigma_u} &\sim \text{Inverse-Wishart}(\xi_{\boldsymbol{u}}, \boldsymbol{\Omega_u}), \\
\sigma^2_{er} &\sim \text{Inverse-Gamma}(\alpha_3, \xi_3).
\end{aligned}
$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p \times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p \times 1$ vector of fixed effect parameters relating to the rth response are denoted by $\boldsymbol{\beta}_r$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha \boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ^2_{er}, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

Spatial correlation in these areal unit level random effects is most often modelled by a conditional autoregressive (CAR) prior distribution. Using this model spatial correlation is induced into the random effects via a non-negative spatial adjacency matrix $\boldsymbol{A} = (a_{sl})_{S \times S}$, which defines how spatially close the S areal units are to each other. The elements of $\boldsymbol{A}_{S \times S}$ can be binary or non-binary, and the most common specification is that $a_{sl} = 1$ if a pair of areal units ($\mathcal{G}_s$, $\mathcal{G}_l$) share a common border or are considered neighbours by some other measure, and $a_{sl} = 0$ otherwise. Note, $a_{ss} = 0$ for all s. τ^2_r measures the variance of these random effects for the rth response, where a conjugate Inverse-Gamma prior is specified for τ^2_r and the corresponding hyperparamaterers (a_1, b_1) can be chosen by the user. ρ_r controls the level of spatial autocorrelation. A non-conjugate uniform prior is specified for ρ_r.

The $R \times 1$ vector of random effects for the jth alter is denoted by $\boldsymbol{u}_j = (u_{j1}, \ldots, u_{jR})_{R \times 1}$, while the $R \times 1$ vector of isolation effects for all R outcomes is denoted by $\boldsymbol{u}^* = (u^*_1, \ldots, u^*_R)$, and both are assigned multivariate Gaussian prior distributions. The unstructured covariance matrix $\boldsymbol{\Sigma_u}$ captures the covariance between the R outcomes at the network level, and a conjugate Inverse-Wishart prior is specified for this covariance matrix $\boldsymbol{\Sigma_u}$. The corresponding hyperparamaterers ($\xi_{\boldsymbol{u}}$, $\boldsymbol{\Omega_u}$) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$\text{Binomial: } Y_{i_sr} \sim \text{Binomial}(n_{i_sr}, \theta_{i_sr}) \text{ and } g(\mu_{i_sr}) = \ln(\theta_{i_sr}/(1-\theta_{i_sr})),$$

$$\text{Gaussian: } Y_{i_sr} \sim \text{N}(\mu_{i_sr}, \sigma^2_{er}) \text{ and } g(\mu_{i_sr}) = \mu_{i_sr},$$

$$\text{Poisson: } Y_{i_sr} \sim \text{Poisson}(\mu_{i_sr}) \text{ and } g(\mu_{i_sr}) = \ln(\mu_{i_sr}).$$

Usage

```
multiNetLeroux(formula, data, trials, family, squareSpatialNeighbourhoodMatrix,
spatialAssignment, W, numberOfSamples = 10, burnin = 0, thin = 1, seed = 1,
trueBeta = NULL, trueSpatialRandomEffects = NULL, trueURandomEffects = NULL,
trueSpatialTauSquared = NULL, trueSpatialRho = NULL,
trueVarianceCovarianceU = NULL, trueSigmaSquaredE = NULL,
covarianceBetaPrior = 10^5, a1 = 0.001, b1 = 0.001, xi, omega, a3 = 0.001,
b3 = 0.001, centerSpatialRandomEffects = TRUE, centerURandomEffects = TRUE)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials n_{i_sr}. Only used if `family`=“binomial".
`family`	The data likelihood model that must be “gaussian", “poisson" or “binomial".
`squareSpatialNeighbourhoodMatrix`	An $S \times S$ symmetric and non-negative neighbourhood matrix $\boldsymbol{A} = (a_{sl})_{S\times S}$.
`W`	A matrix $\boldsymbol{W}$ that encodes the social network structure and whose rows sum to 1.
`spatialAssignment`	The binary matrix of individual's assignment to spatial area used in the model fitting process.
`numberOfSamples`	The number of samples to generate pre-thin.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true value of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_R$.
`trueSpatialRandomEffects`	If available, the true values of $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_R$.
`trueURandomEffects`	If available, the true values of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J, \boldsymbol{u}^*$.
`trueSpatialTauSquared`	If available, the true values of $\tau^2_1, \ldots, \tau^2_R$.

`trueSpatialRho`	If available, the true value of $\rho_1, \ldots, \rho_R$.
`trueVarianceCovarianceU`	If available, the true value of $\boldsymbol{\Sigma}_{\boldsymbol{u}}$.
`trueSigmaSquaredE`	If available, the true value of σ^2_{e1}, …, σ^2_{eR}. Only used if `family`=“gaussian".
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha\boldsymbol{I}$.
`a1`	The shape parameter for the Inverse-Gamma distribution relating to the spatial random effects α_1.
`b1`	The scale parameter for the Inverse-Gamma distribution relating to the spatial random effects ξ_1.
`xi`	The degrees of freedom parameter for the Inverse-Wishart distribution relating to the network random effects $\xi_{\boldsymbol{u}}$.
`omega`	The scale parameter for the Inverse-Wishart distribution relating to the network random effects $\boldsymbol{\Omega}_{\boldsymbol{u}}$.
`a3`	The shape parameter for the Inverse-Gamma distribution relating to the error terms α_3. Only used if `family`=“gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution relating to the error terms ξ_3. Only used if `family`=“gaussian".
`centerSpatialRandomEffects`	A choice to center the spatial random effects after each iteration of the MCMC sampler.
`centerURandomEffects`	A choice to center the network random effects after each iteration of the MCMC sampler.

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`squareSpatialNeighbourhoodMatrix`	The spatial neighbourhood matrix used.
`spatialAssignment`	The spatial assignment matrix used.
`W`	The network matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_R$ parameters in the model.

`spatialTauSquaredSamples`	Type: matrix. The matrix of simulated samples from the posterior distribution of $\tau_1^2, \ldots, \tau_R^2$ in the model.
`spatialRhoSamples`	The vector of simulated samples from the posterior distribution of $\rho_1, \ldots, \rho_R$ in the model.
`varianceCovarianceUSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\Sigma_u}$ in the model.
`spatialRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of spatial random effects $\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_R$ in the model.
`uRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of network random effects $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J, \boldsymbol{u}^*$ in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of $\sigma_{e1}^2, \ldots, \sigma_{eR}^2$ in the model. Only used if `family`=“gaussian".
`acceptanceRates`	The acceptance rates of parameters in the model from the MCMC sampling scheme .
`spatialRandomEffectsAcceptanceRate`	The acceptance rates of spatial random effects in the model from the MCMC sampling scheme.
`uRandomEffectsAcceptanceRate`	The acceptance rates of network random effects in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`DBar`	DBar for the model.
`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

`multiNetRand` *A function that generates samples for a multivariate fixed effects, grouping, and network model.*

Description

This function that generates samples for a multivariate fixed effects, grouping, and network model, which is given by

$$
\begin{aligned}
Y_{i_s r}|\mu_{i_s r} &\sim f(y_{i_s r}|\mu_{i_s r}, \sigma^2_{er}) \quad i = 1, \ldots, N_s,\ s = 1, \ldots, S,\ r = 1, \ldots, R, \\
g(\mu_{i_s r}) &= \boldsymbol{x}_{i_s}^\top \boldsymbol{\beta}_r v_{sr} + \sum_{j \in \text{net}(i_s)} w_{i_s j} u_{jr} + w^*_{i_s} u^*_r, \\
\boldsymbol{\beta}_r &\sim \text{N}(\mathbf{0}, \alpha \boldsymbol{I}) \\
\boldsymbol{v}_s = (v_{s1}, \ldots, v_{sR}) &\sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{v}}) \boldsymbol{v}_s = (v_{s1}, \ldots, v_{sR}) \sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{v}}), \\
\boldsymbol{u}_j = (u_{1j}, \ldots, u_{Rj}) &\sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{u}}), \\
\boldsymbol{u}^* = (u^*_1, \ldots, u^*_R) &\sim \text{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{u}}), \\
\boldsymbol{\Sigma}_{\boldsymbol{v}} &\sim \text{Inverse-Wishart}(\xi_{\boldsymbol{v}}, \boldsymbol{\Omega}_{\boldsymbol{v}}), \\
\boldsymbol{\Sigma}_{\boldsymbol{u}} &\sim \text{Inverse-Wishart}(\xi_{\boldsymbol{u}}, \boldsymbol{\Omega}_{\boldsymbol{u}}), \\
\sigma^2_{er} &\sim \text{Inverse-Gamma}(\alpha_3, \xi_3).
\end{aligned}
$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p \times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p \times 1$ vector of fixed effect parameters relating to the rth response are denoted by $\boldsymbol{\beta}_r$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha \boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ^2_{er}, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

The $R \times 1$ vector of random effects for the \$s\$th group is denoted by $\boldsymbol{v}_s = (v_{s1}, \ldots, v_{sR})_{R \times 1}$, which is assigned a joint Gaussian prior distribution with an unstructured covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{v}}$ that captures the covariance between the R outcomes. A conjugate Inverse-Wishart prior is specified for the random effects covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{v}}$. The corresponding hyperparamaterers ($\xi_{\boldsymbol{v}}$, $\boldsymbol{\Omega}_{\boldsymbol{v}}$) can be chosen by the user.

The $R \times 1$ vector of random effects for the jth alter is denoted by $\boldsymbol{u}_j = (u_{j1}, \ldots, u_{jR})_{R \times 1}$, while the $R \times 1$ vector of isolation effects for all R outcomes is denoted by $\boldsymbol{u}^* = (u^*_1, \ldots, u^*_R)$, and both are assigned multivariate Gaussian prior distributions. The unstructured covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}$ captures the covariance between the R outcomes at the network level, and a conjugate Inverse-Wishart prior is specified for this covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}}$. The corresponding hyperparamaterers ($\xi_{\boldsymbol{u}}$, $\boldsymbol{\Omega}_{\boldsymbol{u}}$) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$
\begin{aligned}
\text{Binomial: } Y_{i_s r} &\sim \text{Binomial}(n_{i_s r}, \theta_{i_s r}) \text{ and } g(\mu_{i_s r}) = \ln(\theta_{i_s r}/(1 - \theta_{i_s r})), \\
\text{Gaussian: } Y_{i_s r} &\sim \text{N}(\mu_{i_s r}, \sigma^2_{er}) \text{ and } g(\mu_{i_s r}) = \mu_{i_s r}, \\
\text{Poisson: } Y_{i_s r} &\sim \text{Poisson}(\mu_{i_s r}) \text{ and } g(\mu_{i_s r}) = \ln(\mu_{i_s r}).
\end{aligned}
$$

Usage

```
multiNetRand(formula, data, trials, family, V, W, numberOfSamples = 10, burnin = 0,
thin = 1, seed = 1, trueBeta = NULL, trueVRandomEffects = NULL,
trueURandomEffects = NULL, trueVarianceCovarianceV = NULL,
trueVarianceCovarianceU = NULL, trueSigmaSquaredE = NULL,
covarianceBetaPrior = 10^5, xiV, omegaV, xi, omega, a3 = 0.001,
b3 = 0.001, centerVRandomEffects = TRUE, centerURandomEffects = TRUE)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials n_{i_sr}. Only used if `family`=“binomial".
`family`	The data likelihood model that must be “gaussian", “poisson" or “binomial".
`V`	The binary matrix of individual’s assignment to groups used in the model fitting process.
`W`	A matrix $\boldsymbol{W}$ that encodes the social network structure and whose rows sum to 1.
`numberOfSamples`	The number of samples to generate pre-thin.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true value of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_R$.
`trueVRandomEffects`	If available, the true values of $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_S$.
`trueURandomEffects`	If available, the true values of $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J, \boldsymbol{u}^*$.
`trueVarianceCovarianceV`	If available, the true value of $\boldsymbol{\Sigma}_{\boldsymbol{v}}$.
`trueVarianceCovarianceU`	If available, the true value of $\boldsymbol{\Sigma}_{\boldsymbol{u}}$.
`trueSigmaSquaredE`	If available, the true value of σ^2_{e1}, . . ., σ^2_{eR}. Only used if `family`=“gaussian".
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha\boldsymbol{I}$.
`xiV`	The degrees of freedom parameter for the Inverse-Wishart distribution relating to the grouping random effects $\xi_{\boldsymbol{v}}$.
`omegaV`	The scale parameter for the Inverse-Wishart distribution relating to the grouping random effects $\boldsymbol{\Omega}_{\boldsymbol{v}}$.

`xi`	The degrees of freedom parameter for the Inverse-Wishart distribution relating to the network random effects $\xi_{\boldsymbol{u}}$.
`omega`	The scale parameter for the Inverse-Wishart distribution relating to the network random effects $\boldsymbol{\Omega_u}$.
`a3`	The shape parameter for the Inverse-Gamma distribution relating to the error terms α_3. Only used if `family`=“gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution relating to the error terms ξ_3. Only used if `family`=“gaussian".
`centerVRandomEffects`	A choice to center the spatial random effects after each iteration of the MCMC sampler.
`centerURandomEffects`	A choice to center the network random effects after each iteration of the MCMC sampler.

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`V`	The grouping assignment matrix used.
`W`	The network matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_R$ parameters in the model.
`varianceCovarianceVSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\Sigma_v}$ in the model.
`varianceCovarianceUSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\Sigma_u}$ in the model.
`vRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of spatial random effects $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_S$ in the model.
`uRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of network random effects $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_J, \boldsymbol{u}^*$ in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of σ^2_{e1}, …, σ^2_{eR} in the model. Only used if `family`=“gaussian".
`acceptanceRates`	The acceptance rates of parameters in the model from the MCMC sampling scheme.

`vRandomEffectsAcceptanceRate`	The acceptance rates of grouping random effects in the model from the MCMC sampling scheme.
`uRandomEffectsAcceptanceRate`	The acceptance rates of network random effects in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`DBar`	DBar for the model.
`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

`plot.netcmc` *A function that plots visual MCMC diagnostics of the fitted model.*

Description

This function takes a netcmc object of samples from the posterior distribution of a parameter(s) and returns a visual convergence diaagnostics in the form of a density plot, trace plot, and ACF plot.

Usage

```
  ## S3 method for class 'netcmc'
plot(x, ...)
```

Arguments

`x`	A netcmc object of samples from the posterior distribution of a parameter(s).
`...`	Ignored.s

Value

Returns a trace plot, density plot and ACF plot for the posterior distribution of a parameter(s) in a netcmc object.

Author(s)

George Gerogiannis

`print.netcmc` *A function that gets a summary of the fitted model.*

Description

This function takes a netcmc object and returns a summary of the fitted model. The summary includes, for selected parameters, posterior medians and 95 percent credible intervals, the effective number of independent samples and the Geweke convergence diagnostic in the form of a Z-score.

Usage

```
  ## S3 method for class 'netcmc'
print(x, ...)
```

Arguments

`x`	A netcmc fitted model object.
`...`	Ignored.s

Value

Returns a model summary for a netcmc object.

Author(s)

George Gerogiannis

`summary.netcmc` *A function that gets a summary of the fitted model.*

Description

This function takes a netcmc object and returns a summary of the fitted model. The summary includes, for selected parameters, posterior medians and 95 percent credible intervals, the effective number of independent samples and the Geweke convergence diagnostic in the form of a Z-score.

Usage

```
  ## S3 method for class 'netcmc'
summary(object, ...)
```

Arguments

`object`	A netcmc fitted model object.
`...`	Ignored.s

Value

Returns a model summary for a netcmc object.

Author(s)

George Gerogiannis

---

uni *A function that generates samples for a univariate fixed effects model.*

Description

This function generates samples for a univariate fixed effects model, which is given by

$$Y_{i_s}|\mu_{i_s} \sim f(y_{i_s}|\mu_{i_s}, \sigma_e^2) \quad i = 1, \ldots, N_s, \; s = 1, \ldots, S,$$

$$g(\mu_{i_s}) = \boldsymbol{x}_{i_s}^\top \boldsymbol{\beta},$$

$$\boldsymbol{\beta} \sim \text{N}(\mathbf{0}, \alpha \boldsymbol{I}),$$

$$\sigma_e^2 \sim \text{Inverse-Gamma}(\alpha_3, \xi_3).$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p \times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p \times 1$ vector of fixed effect parameters are denoted by $\boldsymbol{\beta}$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha \boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ_e^2, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$\text{Binomial: } Y_{i_s} \sim \text{Binomial}(n_{i_s}, \theta_{i_s}) \text{ and } g(\mu_{i_s}) = \ln(\theta_{i_s}/(1-\theta_{i_s})),$$

$$\text{Gaussian: } Y_{i_s} \sim \text{N}(\mu_{i_s}, \sigma_e^2) \text{ and } g(\mu_{i_s}) = \mu_{i_s},$$

$$\text{Poisson: } Y_{i_s} \sim \text{Poisson}(\mu_{i_s}) \text{ and } g(\mu_{i_s}) = \ln(\mu_{i_s}).$$

Usage

```
uni(formula, data, trials, family, numberOfSamples = 10, burnin = 0, thin = 1, seed = 1,
trueBeta = NULL, trueSigmaSquaredE = NULL, covarianceBetaPrior = 10^5,
a3 = 0.001, b3 = 0.001)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials n_{i_s}. Only used if `family`="binomial".
`family`	The data likelihood model that must be "gaussian" , "poisson" or "binomial".
`numberOfSamples`	The number of samples to generate pre-thin.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true values of the $\boldsymbol{\beta}$.
`trueSigmaSquaredE`	If available, the true value of σ^2_e. Only used if `family`="gaussian".
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha \boldsymbol{I}$.
`a3`	The shape parameter for the Inverse-Gamma distribution α_3. Only used if `family`="gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution ξ_3. Only used if `family`="gaussian".

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}$ parameters in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of σ^2_e in the model.
`acceptanceRates`	The acceptance rates of parameters in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`DBar`	DBar for the model.

`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

Examples

```
  ##################################################
  #### Run the model on simulated data
  ##################################################

  #### Generate the covariates and response data
  observations <- 100
  X <- matrix(rnorm(2 * observations), ncol = 2)
  colnames(X) <- c("x1", "x2")
  beta <- c(2, -2, 2)
  logit <- cbind(rep(1, observations), X) %*% beta
  prob <- exp(logit) / (1 + exp(logit))
  trials <- rep(50, observations)
  Y <- rbinom(n = observations, size = trials, prob = prob)
  data <- data.frame(cbind(Y, X))

  #### Run the model
  formula <- Y ~ x1 + x2
  ## Not run: model <- uni(formula = formula, data = data, family="binomial",
                        trials = trials, numberOfSamples = 10000,
                        burnin = 10000, thin = 10, seed = 1)
## End(Not run)
```

`uniNet` *A function that generates samples for a univariate network model.*

Description

This function generates samples for a univariate network model, which is given by

$$
\begin{gathered}
Y_{i_s}|\mu_{i_s} \sim f(y_{i_s}|\mu_{i_s}, \sigma_e^2) \quad i = 1, \dots, N_s,\ s = 1, \dots, S, \\
g(\mu_{i_s}) = \boldsymbol{x}_{i_s}^\top \boldsymbol{\beta} + \sum_{j \in \text{net}(i_s)} w_{i_s j} u_j + w_{i_s}^* u^*, \\
\boldsymbol{\beta} \sim \mathrm{N}(\mathbf{0}, \alpha \boldsymbol{I}),
\end{gathered}
$$

$$
\begin{aligned}
u_j &\sim \mathrm{N}(0, \sigma_u^2),\\
u^* &\sim \mathrm{N}(0, \sigma_u^2),\\
\sigma_u^2 &\sim \text{Inverse-Gamma}(\alpha_2, \xi_2),\\
\sigma_e^2 &\sim \text{Inverse-Gamma}(\alpha_3, \xi_3).
\end{aligned}
$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p \times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p \times 1$ vector of fixed effect parameters are denoted by $\boldsymbol{\beta}$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha \boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ_e^2, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

The $J \times 1$ vector of alter random effects are denoted by $\boldsymbol{u} = (u_1, \ldots, u_J)_{J \times 1}$ and modelled as independently Gaussian with mean zero and a constant variance, and due to the row standardised nature of $\boldsymbol{W}$, $\sum_{j \in \text{net}(i_s)} w_{i_s j} u_j$ represents the average (mean) effect that the peers of individual i in spatial unit or group s have on that individual. $w_{i_s}^* u^*$ is an *isolation effect*, which is an effect for individuals who don't nominate any friends. This is achieved by setting $w_{i_s}^* = 1$ if individual i_s nominates no peers and $w_{i_s}^* = 0$ otherwise, and if $w_{i_s}^* = 1$ then clearly $\sum_{j \in \text{net}(i_s)} w_{i_s j} u_{jr} = 0$ as net(i_s) is the empty set. A conjugate Inverse-Gamma prior is specified for the random effects variance σ_u^2, and the corresponding hyperparamaterers (α_2, ξ_2) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$
\begin{aligned}
\text{Binomial: } Y_{i_s} &\sim \text{Binomial}(n_{i_s}, \theta_{i_s}) \text{ and } g(\mu_{i_s}) = \ln(\theta_{i_s}/(1-\theta_{i_s})),\\
\text{Gaussian: } Y_{i_s} &\sim \mathrm{N}(\mu_{i_s}, \sigma_e^2) \text{ and } g(\mu_{i_s}) = \mu_{i_s},\\
\text{Poisson: } Y_{i_s} &\sim \text{Poisson}(\mu_{i_s}) \text{ and } g(\mu_{i_s}) = \ln(\mu_{i_s}).
\end{aligned}
$$

Usage

```
uniNet(formula, data, trials, family, W, numberOfSamples = 10, burnin = 0, thin = 1,
seed = 1, trueBeta = NULL, trueURandomEffects = NULL, trueSigmaSquaredU = NULL,
trueSigmaSquaredE = NULL, covarianceBetaPrior = 10^5, a2 = 0.001, b2 = 0.001,
a3 = 0.001, b3 = 0.001, centerURandomEffects = TRUE)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials n_{i_s}. Only used if `family`="binomial".
`family`	The data likelihood model that must be "gaussian", "poisson" or "binomial".
`W`	A matrix $\boldsymbol{W}$ that encodes the social network structure and whose rows sum to 1.
`numberOfSamples`	The number of samples to generate pre-thin.

`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true value of $\boldsymbol{\beta}$.
`trueURandomEffects`	If available, the true value of $\boldsymbol{u}$.
`trueSigmaSquaredU`	If available, the true value σ_u^2.
`trueSigmaSquaredE`	If available, the true value σ_e^2.
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha\boldsymbol{I}$.
`a2`	The shape parameter for the Inverse-Gamma distribution relating to the network random effects α_2.
`b2`	The scale parameter for the Inverse-Gamma distribution relating to the network random effects ξ_2.
`a3`	The shape parameter for the Inverse-Gamma distribution relating to the error terms α_3. Only used if `family`=“gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution relating to the error terms ξ_3. Only used if `family`=“gaussian".
`centerURandomEffects`	A choice to center the network random effects after each iteration of the MCMC sampler.

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`W`	The network matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}$ parameters in the model.
`sigmaSquaredUSamples`	The vector of simulated samples from the posterior distribution of σ_u^2 in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of σ_e^2 in the model.

`uRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of network random effects $\boldsymbol{u}$ in the model.
`acceptanceRates`	The acceptance rates of parameters in the model (excluding random effects) from the MCMC sampling scheme .
`uRandomEffectsAcceptanceRate`	The acceptance rates of network random effects in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`DBar`	DBar for the model.
`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

Examples

```
##################################################
#### Run the model on simulated data
##################################################
#### Load other libraries required
library(MCMCpack)

#### Set up a network
observations <- 200
numberOfMultipleClassifications <- 50
W <- matrix(rbinom(observations * numberOfMultipleClassifications, 1, 0.05),
            ncol = numberOfMultipleClassifications)
numberOfActorsWithNoPeers <- sum(apply(W, 1, function(x) { sum(x) == 0 }))
peers <- sample(1:numberOfMultipleClassifications, numberOfActorsWithNoPeers,
                TRUE)
actorsWithNoPeers <- which(apply(W, 1, function(x) { sum(x) == 0 }))
for(i in 1:numberOfActorsWithNoPeers) {
  W[actorsWithNoPeers[i], peers[i]] <- 1
}
W <- t(apply(W, 1, function(x) { x / sum(x) }))

#### Generate the covariates and response data
X <- matrix(rnorm(2 * observations), ncol = 2)
colnames(X) <- c("x1", "x2")
```

```
  beta <- c(1, -0.5, 0.5)
  sigmaSquaredU <- 1
  uRandomEffects <- rnorm(numberOfMultipleClassifications, mean = 0,
                          sd = sqrt(sigmaSquaredU))

  logTheta <- cbind(rep(1, observations), X) %*% beta + W %*% uRandomEffects
  Y <- rpois(n = observations, lambda = exp(logTheta))
  data <- data.frame(cbind(Y, X))

  #### Run the model
  formula <- Y ~ x1 + x2
  ## Not run: model <- uniNet(formula = formula, data = data, family="poisson",
                          W = W, numberOfSamples = 10000, burnin = 10000,
                          thin = 10, seed = 1)
## End(Not run)
```

`uniNetLeroux` *A function that generates samples for a univariate network Leroux model.*

Description

This function generates samples for a univariate network Leroux model, which is given by

$$\begin{aligned}
Y_{i_s}|\mu_{i_s} &\sim f(y_{i_s}|\mu_{i_s},\sigma_e^2) \quad i=1,\ldots,N_s,\ s=1,\ldots,S,\\
g(\mu_{i_s}) &= \boldsymbol{x}_{i_s}^\top\boldsymbol{\beta}+\phi_s+\sum_{j\in\text{net}(i_s)} w_{i_sj}u_j + w_{i_s}^*u^*,\\
\boldsymbol{\beta} &\sim \mathrm{N}(\mathbf{0},\alpha\boldsymbol{I}),\\
\phi_s|\boldsymbol{\phi}_{-s} &\sim \mathrm{N}\left(\frac{\rho\sum_{l=1}^S a_{sl}\phi_l}{\rho\sum_{l=1}^S a_{sl}+1-\rho},\frac{\tau^2}{\rho\sum_{l=1}^S a_{sl}+1-\rho}\right),\\
u_j &\sim \mathrm{N}(0,\sigma_u^2),\\
u^* &\sim \mathrm{N}(0,\sigma_u^2),\\
\tau^2 &\sim \text{Inverse-Gamma}(\alpha_1,\xi_1),\\
\rho &\sim \text{Uniform}(0,1),\\
\sigma_u^2 &\sim \text{Inverse-Gamma}(\alpha_2,\xi_2),\\
\sigma_e^2 &\sim \text{Inverse-Gamma}(\alpha_3,\xi_3).
\end{aligned}$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p\times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p\times 1$ vector of fixed effect parameters are denoted by $\boldsymbol{\beta}$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha\boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ_e^2, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

Spatial correlation in these areal unit level random effects is most often modelled by a conditional autoregressive (CAR) prior distribution. Using this model spatial correlation is induced into the random effects via a non-negative spatial adjacency matrix $\boldsymbol{A} = (a_{sl})_{S\times S}$, which defines how spatially close the S areal units are to each other. The elements of $\boldsymbol{A}_{S\times S}$ can be binary or non-binary, and the most common specification is that $a_{sl} = 1$ if a pair of areal units ($\mathcal{G}_s$, $\mathcal{G}_l$) share a common border or are considered neighbours by some other measure, and $a_{sl} = 0$ otherwise. Note, $a_{ss} = 0$ for all s. $\boldsymbol{\phi}_{-s} = (\phi_1, \ldots, \phi_{s-1}, \phi_{s+1}, \ldots, \phi_S)$. Here τ^2 is a measure of the variance relating to the spatial random effects $\boldsymbol{\phi}$, while ρ controls the level of spatial autocorrelation, with values close to one and zero representing strong autocorrelation and independence respectively. A non-conjugate uniform prior on the unit interval is specified for the single level of spatial autocorrelation ρ. In contrast, a conjugate Inverse-Gamma prior is specified for the random effects variance τ^2, and corresponding hyperparamaterers (α_1, ξ_1) can be chosen by the user.

The $J \times 1$ vector of alter random effects are denoted by $\boldsymbol{u} = (u_1, \ldots, u_J)_{J\times 1}$ and modelled as independently Gaussian with mean zero and a constant variance, and due to the row standardised nature of $\boldsymbol{W}$, $\sum_{j\in \text{net}(i_s)} w_{i_s j} u_j$ represents the average (mean) effect that the peers of individual i in spatial unit or group s have on that individual. $w^*_{i_s} u^*$ is an *isolation effect*, which is an effect for individuals who don't nominate any friends. This is achieved by setting $w^*_{i_s} = 1$ if individual i_s nominates no peers and $w^*_{i_s} = 0$ otherwise, and if $w^*_{i_s} = 1$ then clearly $\sum_{j\in \text{net}(i_s)} w_{i_s j} u_{jr} = 0$ as net(i_s) is the empty set. A conjugate Inverse-Gamma prior is specified for the random effects variance σ^2_u, and the corresponding hyperparamaterers (α_2, ξ_2) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$\text{Binomial: } Y_{i_s} \sim \text{Binomial}(n_{i_s}, \theta_{i_s}) \text{ and } g(\mu_{i_s}) = \text{ln}(\theta_{i_s}/(1-\theta_{i_s})),$$

$$\text{Gaussian: } Y_{i_s} \sim \text{N}(\mu_{i_s}, \sigma^2_e) \text{ and } g(\mu_{i_s}) = \mu_{i_s},$$

$$\text{Poisson: } Y_{i_s} \sim \text{Poisson}(\mu_{i_s}) \text{ and } g(\mu_{i_s}) = \text{ln}(\mu_{i_s}).$$

Usage

```
uniNetLeroux(formula, data, trials, family,
squareSpatialNeighbourhoodMatrix, spatialAssignment, W, numberOfSamples = 10,
burnin = 0, thin = 1, seed = 1, trueBeta = NULL,
trueSpatialRandomEffects = NULL, trueURandomEffects = NULL,
trueSpatialTauSquared = NULL, trueSpatialRho = NULL, trueSigmaSquaredU = NULL,
trueSigmaSquaredE = NULL, covarianceBetaPrior = 10^5, a1 = 0.001, b1 = 0.001,
a2 = 0.001, b2 = 0.001, a3 = 0.001, b3 = 0.001,
centerSpatialRandomEffects = TRUE, centerURandomEffects = TRUE)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials n_{i_s}. Only used if `family`=“binomial".
`family`	The data likelihood model that must be “gaussian", “poisson" or “binomial".

`squareSpatialNeighbourhoodMatrix`	An $S \times S$ symmetric and non-negative neighbourhood matrix $\boldsymbol{A} = (a_{sl})_{S \times S}$.
`W`	A matrix $\boldsymbol{W}$ that encodes the social network structure and whose rows sum to 1.
`spatialAssignment`	The binary matrix of individual's assignment to spatial area used in the model fitting process.
`numberOfSamples`	The number of samples to generate pre-thin.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true value of $\boldsymbol{\beta}$.
`trueSpatialRandomEffects`	If available, the true value of $\boldsymbol{\phi}$.
`trueURandomEffects`	If available, the true value of $\boldsymbol{u}$.
`trueSpatialTauSquared`	If available, the true value of τ^2.
`trueSpatialRho`	If available, the true value ofρ.
`trueSigmaSquaredU`	If available, the true value of σ^2_u.
`trueSigmaSquaredE`	If available, the true value of σ^2_e.
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha\boldsymbol{I}$.
`a1`	The shape parameter for the Inverse-Gamma distribution relating to the spatial random effects α_1.
`b1`	The scale parameter for the Inverse-Gamma distribution relating to the spatial random effects ξ_1.
`a2`	The shape parameter for the Inverse-Gamma distribution relating to the network random effects α_2.
`b2`	The scale parameter for the Inverse-Gamma distribution relating to the network random effects ξ_2.
`a3`	The shape parameter for the Inverse-Gamma distribution relating to the error terms α_3. Only used if `family`=“gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution relating to the error terms ξ_3. Only used if `family`=“gaussian".
`centerSpatialRandomEffects`	A choice to center the spatial random effects after each iteration of the MCMC sampler.
`centerURandomEffects`	A choice to center the network random effects after each iteration of the MCMC sampler.

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`squareSpatialNeighbourhoodMatrix`	The spatial neighbourhood matrix used.
`spatialAssignment`	The spatial assignment matrix used.
`W`	The network matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}$ parameters in the model.
`spatialTauSquaredSamples`	The vector of simulated samples from the posterior distribution of τ^2 in the model.
`spatialRhoSamples`	The vector of simulated samples from the posterior distribution of ρ in the model.
`sigmaSquaredUSamples`	The vector of simulated samples from the posterior distribution of σ_u^2 in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of σ_e^2 in the model.
`spatialRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of spatial/grouping random effects $\boldsymbol{\phi}$ in the model.
`uRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of network random effects $\boldsymbol{u}$ in the model.
`acceptanceRates`	The acceptance rates of parameters in the model (excluding random effects) from the MCMC sampling scheme .
`spatialRandomEffectsAcceptanceRate`	The acceptance rates of spatial/grouping random effects in the model from the MCMC sampling scheme.
`uRandomEffectsAcceptanceRate`	The acceptance rates of network random effects in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.

`DBar`	DBar for the model.
`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

Examples

```
#################################################
#### Run the model on simulated data
#################################################
#### Load other libraries required
library(MCMCpack)

#### Set up a network
observations <- 200
numberOfMultipleClassifications <- 50
W <- matrix(rbinom(observations * numberOfMultipleClassifications, 1, 0.05),
            ncol = numberOfMultipleClassifications)
numberOfActorsWithNoPeers <- sum(apply(W, 1, function(x) { sum(x) == 0 }))
peers <- sample(1:numberOfMultipleClassifications, numberOfActorsWithNoPeers,
TRUE)
actorsWithNoPeers <- which(apply(W, 1, function(x) { sum(x) == 0 }))
for(i in 1:numberOfActorsWithNoPeers) {
  W[actorsWithNoPeers[i], peers[i]] <- 1
}
W <- t(apply(W, 1, function(x) { x / sum(x) }))

#### Set up a spatial structure
numberOfSpatialAreas <- 100
factor = sample(1:numberOfSpatialAreas, observations, TRUE)
spatialAssignment = matrix(NA, ncol = numberOfSpatialAreas,
                           nrow = observations)
for(i in 1:length(factor)){
  for(j in 1:numberOfSpatialAreas){
    if(factor[i] == j){
      spatialAssignment[i, j] = 1
    } else {
      spatialAssignment[i, j] = 0
    }
  }
}

gridAxis = sqrt(numberOfSpatialAreas)
easting = 1:gridAxis
```

```
    northing = 1:gridAxis
    grid = expand.grid(easting, northing)
    numberOfRowsInGrid = nrow(grid)
    distance = as.matrix(dist(grid))
    squareSpatialNeighbourhoodMatrix = array(0, c(numberOfRowsInGrid,
                                                  numberOfRowsInGrid))
    squareSpatialNeighbourhoodMatrix[distance==1] = 1

    #### Generate the covariates and response data
    X <- matrix(rnorm(2 * observations), ncol = 2)
    colnames(X) <- c("x1", "x2")
    beta <- c(2, -2, 2)

    spatialRho <- 0.5
    spatialTauSquared <- 2
    spatialPrecisionMatrix = spatialRho *
      (diag(apply(squareSpatialNeighbourhoodMatrix, 1, sum)) -
       squareSpatialNeighbourhoodMatrix) + (1 - spatialRho) *
       diag(rep(1, numberOfSpatialAreas))
    spatialCovarianceMatrix = solve(spatialPrecisionMatrix)
    spatialPhi = mvrnorm(n = 1, mu = rep(0, numberOfSpatialAreas),
                         Sigma = (spatialTauSquared * spatialCovarianceMatrix))

    sigmaSquaredU <- 2
    uRandomEffects <- rnorm(numberOfMultipleClassifications, mean = 0,
                            sd = sqrt(sigmaSquaredU))

    logit <- cbind(rep(1, observations), X) %*% beta +
      spatialAssignment %*% spatialPhi + W %*% uRandomEffects
    prob <- exp(logit) / (1 + exp(logit))
    trials <- rep(50, observations)
    Y <- rbinom(n = observations, size = trials, prob = prob)
    data <- data.frame(cbind(Y, X))

    #### Run the model
    formula <- Y ~ x1 + x2
    ## Not run: model <- uniNetLeroux(formula = formula, data = data,
      family="binomial",  W = W,
      spatialAssignment = spatialAssignment,
      squareSpatialNeighbourhoodMatrix = squareSpatialNeighbourhoodMatrix,
      trials = trials, numberOfSamples = 10000,
      burnin = 10000, thin = 10, seed = 1)
  ## End(Not run)
```

`uniNetRand` *A function that generates samples for a univariate network group model.*

Description

This function generates samples for a univariate network group model, which is given by

$$
\begin{aligned}
Y_{i_s}|\mu_{i_s} &\sim f(y_{i_s}|\mu_{i_s},\sigma_e^2) \quad i=1,\ldots,N_s,\ s=1,\ldots,S,\\
g(\mu_{i_s}) &= \boldsymbol{x}_{i_s}^\top\boldsymbol{\beta} + v_s + \sum_{j\in \text{net}(i_s)} w_{i_s j}u_j + w_{i_s}^* u^*,\\
\boldsymbol{\beta} &\sim \text{N}(\mathbf{0},\alpha\boldsymbol{I}),\\
v_s &\sim \text{N}(0,\tau^2),\\
u_j &\sim \text{N}(0,\sigma_u^2),\\
u^* &\sim \text{N}(0,\sigma_u^2),\\
\tau^2 &\sim \text{Inverse-Gamma}(\alpha_1,\xi_1),\\
\sigma_u^2 &\sim \text{Inverse-Gamma}(\alpha_2,\xi_2),\\
\sigma_e^2 &\sim \text{Inverse-Gamma}(\alpha_3,\xi_3).
\end{aligned}
$$

The covariates for the ith individual in the sth spatial unit or other grouping are included in a $p \times 1$ vector $\boldsymbol{x}_{i_s}$. The corresponding $p \times 1$ vector of fixed effect parameters are denoted by $\boldsymbol{\beta}$, which has an assumed multivariate Gaussian prior with mean $\mathbf{0}$ and diagonal covariance matrix $\alpha\boldsymbol{I}$ that can be chosen by the user. A conjugate Inverse-Gamma prior is specified for σ_e^2, and the corresponding hyperparamaterers (α_3, ξ_3) can be chosen by the user.

The $S \times 1$ vector of random effects for the groups are collectively denoted by $\boldsymbol{v} = (v_1,\ldots,v_S)_{S\times 1}$, and each element is assigned an independent zero-mean Gaussian prior distribution with a constant variance τ^2. A conjugate Inverse-Gamma prior is specified for τ^2. The corresponding hyperparamaterers (α_1, ξ_1) can be chosen by the user.

The $J \times 1$ vector of alter random effects are denoted by $\boldsymbol{u} = (u_1,\ldots,u_J)_{J\times 1}$ and modelled as independently Gaussian with mean zero and a constant variance, and due to the row standardised nature of $\boldsymbol{W}$, $\sum_{j\in \text{net}(i_s)} w_{i_s j}u_j$ represents the average (mean) effect that the peers of individual i in spatial unit or group s have on that individual. $w_{i_s}^* u^*$ is an *isolation effect*, which is an effect for individuals who don't nominate any friends. This is achieved by setting $w_{i_s}^* = 1$ if individual i_s nominates no peers and $w_{i_s}^* = 0$ otherwise, and if $w_{i_s}^* = 1$ then clearly $\sum_{j\in \text{net}(i_s)} w_{i_s j}u_{jr} = 0$ as net(i_s) is the empty set. A conjugate Inverse-Gamma prior is specified for the random effects variance σ_u^2, and the corresponding hyperparamaterers (α_2, ξ_2) can be chosen by the user.

The exact specification of each of the likelihoods (binomial, Gaussian, and Poisson) are given below:

$$
\begin{aligned}
\text{Binomial: } Y_{i_s} &\sim \text{Binomial}(n_{i_s},\theta_{i_s}) \text{ and } g(\mu_{i_s}) = \text{ln}(\theta_{i_s}/(1-\theta_{i_s})),\\
\text{Gaussian: } Y_{i_s} &\sim \text{N}(\mu_{i_s},\sigma_e^2) \text{ and } g(\mu_{i_s}) = \mu_{i_s},\\
\text{Poisson: } Y_{i_s} &\sim \text{Poisson}(\mu_{i_s}) \text{ and } g(\mu_{i_s}) = \text{ln}(\mu_{i_s}).
\end{aligned}
$$

Usage

```
uniNetRand(formula, data, trials, family, groupAssignment, W, numberOfSamples = 10,
burnin = 0, thin = 1, seed = 1, trueBeta = NULL,
trueGroupRandomEffects = NULL, trueURandomEffects = NULL,
trueTauSquared = NULL, trueSigmaSquaredU = NULL,
trueSigmaSquaredE = NULL, covarianceBetaPrior = 10^5, a1 = 0.001, b1 = 0.001,
a2 = 0.001, b2 = 0.001, a3 = 0.001, b3 = 0.001,
centerGroupRandomEffects = TRUE, centerURandomEffects = TRUE)
```

Arguments

`formula`	A formula for the covariate part of the model using a similar syntax to that used in the lm() function.
`data`	An optional data.frame containing the variables in the formula.
`trials`	A vector the same length as the response containing the total number of trials n_{i_s}. Only used if `family`=“binomial".
`family`	The data likelihood model that must be “gaussian", “poisson" or “binomial".
`W`	A matrix $\boldsymbol{W}$ that encodes the social network structure and whose rows sum to 1.
`groupAssignment`	The binary matrix of individual’s assignment to groups used in the model fitting process.
`numberOfSamples`	The number of samples to generate pre-thin.
`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`seed`	A seed for the MCMC algorithm.
`trueBeta`	If available, the true value of $\boldsymbol{\beta}$.
`trueGroupRandomEffects`	If available, the true value of $\boldsymbol{v}$.
`trueURandomEffects`	If available, the true value of $\boldsymbol{u}$.
`trueTauSquared`	If available, the true value τ^2.
`trueSigmaSquaredU`	If available, the true value σ^2_u.
`trueSigmaSquaredE`	If available, the true value σ^2_e.
`covarianceBetaPrior`	A scalar prior α for the covariance parameter of the beta prior, such that the covariance is $\alpha\boldsymbol{I}$.
`a1`	The shape parameter for the Inverse-Gamma distribution relating to the group random effects α_1.
`b1`	The shape parameter for the Inverse-Gamma distribution relating to the group random effects ξ_1.
`a2`	The shape parameter for the Inverse-Gamma distribution relating to the network random effects α_2.
`b2`	The scale parameter for the Inverse-Gamma distribution relating to the network random effects ξ_2.
`a3`	The shape parameter for the Inverse-Gamma distribution relating to the error terms α_3. Only used if `family`=“gaussian".
`b3`	The scale parameter for the Inverse-Gamma distribution relating to the error terms ξ_3. Only used if `family`=“gaussian".

`centerGroupRandomEffects`	A choice to center the group random effects after each iteration of the MCMC sampler.
`centerURandomEffects`	A choice to center the network random effects after each iteration of the MCMC sampler.

Value

`call`	The matched call.
`y`	The response used.
`X`	The design matrix used.
`standardizedX`	The standardized design matrix used.
`groupAssignment`	The group assignment matrix used.
`W`	The network matrix used.
`samples`	The matrix of simulated samples from the posterior distribution of each parameter in the model (excluding random effects).
`betaSamples`	The matrix of simulated samples from the posterior distribution of $\boldsymbol{\beta}$ parameters in the model.
`tauSquaredSamples`	The vector of simulated samples from the posterior distribution of τ^2 in the model.
`sigmaSquaredUSamples`	The vector of simulated samples from the posterior distribution of σ_u^2 in the model.
`sigmaSquaredESamples`	The vector of simulated samples from the posterior distribution of σ_e^2 in the model.
`groupRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of spatial/grouping random effects $\boldsymbol{v}$ in the model.
`uRandomEffectsSamples`	The matrix of simulated samples from the posterior distribution of network random effects $\boldsymbol{u}$ in the model.
`acceptanceRates`	The acceptance rates of parameters in the model (excluding random effects) from the MCMC sampling scheme .
`groupRandomEffectsAcceptanceRate`	The acceptance rates of spatial/grouping random effects in the model from the MCMC sampling scheme.
`uRandomEffectsAcceptanceRate`	The acceptance rates of network random effects in the model from the MCMC sampling scheme.
`timeTaken`	The time taken for the model to run.

`burnin`	The number of MCMC samples to discard as the burn-in period.
`thin`	The value by which to thin `numberOfSamples`.
`DBar`	DBar for the model.
`posteriorDeviance`	The posterior deviance for the model.
`posteriorLogLikelihood`	The posterior log likelihood for the model.
`pd`	The number of effective parameters in the model.
`DIC`	The DIC for the model.

Author(s)

George Gerogiannis

Examples

```
#################################################
#### Run the model on simulated data
#################################################
#### Load other libraries required
library(MCMCpack)

#### Set up a network
observations <- 200
numberOfMultipleClassifications <- 50
W <- matrix(rbinom(observations * numberOfMultipleClassifications, 1, 0.05),
            ncol = numberOfMultipleClassifications)
numberOfActorsWithNoPeers <- sum(apply(W, 1, function(x) { sum(x) == 0 }))
peers <- sample(1:numberOfMultipleClassifications, numberOfActorsWithNoPeers,
                TRUE)
actorsWithNoPeers <- which(apply(W, 1, function(x) { sum(x) == 0 }))
for(i in 1:numberOfActorsWithNoPeers) {
  W[actorsWithNoPeers[i], peers[i]] <- 1
}
W <- t(apply(W, 1, function(x) { x / sum(x) }))

#### Set up a single level classification
numberOfSingleClassifications <- 20
factor = sample(1:numberOfSingleClassifications, observations, TRUE)
V = matrix(NA, ncol = numberOfSingleClassifications, nrow = observations)
for(i in 1:length(factor)){
  for(j in 1:numberOfSingleClassifications){
    if(factor[i] == j){
      V[i, j] = 1
    } else {
      V[i, j] = 0
    }
  }
}
```

```
    #### Generate the covariates and response data
    X <- matrix(rnorm(2 * observations), ncol = 2)
    colnames(X) <- c("x1", "x2")
    beta <- c(1, -0.5, 0.5)
    tauSquared <- 0.5
    vRandomEffects <- rnorm(numberOfSingleClassifications, mean = 0,
                            sd = sqrt(tauSquared))
    sigmaSquaredU <- 1
    uRandomEffects <- rnorm(numberOfMultipleClassifications, mean = 0,
                            sd = sqrt(sigmaSquaredU))

    logTheta <- cbind(rep(1, observations), X) %*% beta + V %*% vRandomEffects
              + W %*% uRandomEffects
    Y <- rpois(n = observations, lambda = exp(logTheta))
    data <- data.frame(cbind(Y, X))

    #### Run the model
    formula <- Y ~ x1 + x2
    ## Not run: model <- uniNetRand(formula = formula, data = data, family="poisson",
                                W = W, groupAssignment = V,
                                numberOfSamples = 10000, burnin = 10000,
                                thin = 10, seed = 1)
  ## End(Not run)
```

Index

getAdjacencyMatrix, 3
getMembershipMatrix, 5
getTotalAltersByStatus, 6

multiNet, 7
multiNetLeroux, 10
multiNetRand, 14

netcmc *(*netcmc-package*)*, 2
netcmc-package, 2

plot.netcmc, 17
print.netcmc, 18

summary.netcmc, 18

uni, 19
uniNet, 21
uniNetLeroux, 25
uniNetRand, 30