Package ‘glmmBUGS’

February 19, 2015

Type Package

Title Generalised Linear Mixed Models and Spatial Models with WinBUGS, BRugs, or OpenBUGS

Version 2.3

Date 2013-11-26

Depends R (>= 3.0.0)

Imports MASS, abind,

Suggests spdep

Enhances R2WinBUGS, R2OpenBUGS, BRugs, R2jags, sp, nlme

Author Patrick Brown

Maintainer Patrick Brown <patrick.brown@utoronto.ca>

Description Write bugs model files for hierarchical and spatial models, arranges unbalanced data in ragged arrays, and creates starting values.

License GPL

Repository CRAN

Repository/R-Forge/Project glmmbugs

Repository/R-Forge/Revision 254

Repository/R-Forge/DateTimeStamp 2013-12-19 20:46:40

Date/Publication 2014-01-02 23:12:37

NeedsCompilation no

R topics documented:

  addSpatial .................................................... 2
  binToBinom .................................................. 3
  checkChain ..................................................... 4
  cholInvArray ............................................... 5
  getDesignMatrix .......................................... 6
  getRaggedSeq ............................................. 7
  getStartingValues ....................................... 8
**addSpatial**

Calculate adjacency values for WinBUGS

**Description**

Put an adjacency object in a ragged array

**Usage**

```
addSpatial(map, raggedArray, effect = NULL, prefix=NULL)
```

**Arguments**

- `map`: a spatialPolygonsDataFrame object, or an nb object or a list of two vectors, `adj` and `num`
- `raggedArray`: the result from `winBugsRaggedArray`
- `effect`: a character vector listing the effect names
- `prefix`: Character string to be appended to variable names

**Details**

Computes the values needed by the `car.normal` distribution in WinBUGS. This function is called by `glmmBUGS` when a spatial argument is provided, `addSpatial` is usually not called by a user.

**Value**

The ragged array is returned, with the following additional elements

- `num`: a vector of the number of neighbours of each region
- `adj`: a vector containing the neighbours
- `weights`: a vector of ones, the same length as `adj`
- `NregionSpatial`: where 'region' is replaced by the name of the effect. The number of regions.
binToBinom

Author(s)

Patrick Brown

References

Also see the geoBUGS manual

Examples

```r
## Not run:

# get a winbugs model and data ready, without a spatial effect
data(ontario)

forBugs = glmmBUGS(formula=observed + logExpected ~ 1,
effects="CSDUID", family="poisson",
data=data.frame(ontario))

# now add a spatial effect.
# first, compute the adjacency matrix
# if region ID's are stored as factors, make sure to convert
# them to characters rather than the default of converting them
# to integers
library(diseasemapping)
data(popdata)
popDataAdjMat = poly2nb(popdata, row.names=as.character(popdata[["CSDUID"]]))
data(popDataAdjMat)

# add the adjacency matrix to the ragged array
raggedWithSpatial = addSpatial(popDataAdjMat, forBugs$ragged, "CSDUID")

# write a new bugs model with a spatial effect
writeBugsModel("model.bug", "CSDUID", NULL, c("count", "expected"),
"poisson", spatial="CSDUID")
startingValues = forBugs$startingValues
source("getInits.R")

library(R2WinBUGS)
popResult = bugs(raggedWithSpatial, getInits,
parameters.to.save = names(getInits()), model.file="model.bug",
chain=3, n.iter=1000, n.burnin=100, n.thin=10, program="winbugs")

## End(Not run)
```

binToBinom Convert Bernoulli observations to Binomial
checkChain

Description
Combines multiple Bernoulli observations with the same covariates into one Binomial response

Usage
binToBinom(obs, covariates)

Arguments
obs logical vector of observations
covariates Data frame or matrix of covariates

Value
A data frame with one row for each unique value for the covariates, including the covariates and the following additional columns:
y Number of positive observations for the corresponding covariate values
N Total number of observations for these covariates

Author(s)
Patrick Brown

Examples
thedata = data.frame(sex = rep(c("m", "f"), 10), age=rep(c(20,30), c(10, 10)))
y = rbinom(dim(thedata)[1], 1, 0.5)
bindata = binToBinom(y, thedata)
bindata$zeros = bindata$N - bindata$y
glm(as.matrix(bindata[,c("y", "zeros")]) ~ sex, data=bindata, family=binomial)

checkChain

Plot an MCMC run

Description
Makes time series plots of the parameters (not the random effects) of an MCMC run.

Usage
checkChain(chain, parameters=NULL, oneFigure=TRUE)
cholInvArray

Arguments

chain The result from restoreParams, or the sims.array component of a bugs call.
parameters Vector of character strings giving names of parameters to plot. Default is all parameters with names starting with either "beta", "intercept", or "SD".
oneFigure if TRUE, use par(mfrow=c(a, b)) to put all plots on the same device. Otherwise create a new device for each plot.

Value

Plots are produced, nothing is returned

Author(s)

Patrick Brown

See Also

restoreParams, summaryChain

Examples

thechain = list(beta = array(1, c(10, 3, 4)),
dimnames = list(NULL, NULL, paste("beta", 1:4, "", sep=""))),
  intercept = matrix(1, 10, 3))

checkChain(thechain)

cholInvArray

Precision matrices to variance matrices for Winbugs output

Description

Given an array containing simulations from the posterior of a precision matrix, each individual precision matrix is converted to variances, covariances, and correlations.

Usage

cholInvArray(x, prefix = "T", chol=FALSE)

Arguments

x An array of winbugs output, with precision matrix entries of the form "T[1,3]"
prefix The name of the precision matrix in winbugs, the "T" in "T[1,2]"
chol If TRUE, the cholesky decomposition is returned instead of the inverse
getDesignMatrix

Computes a design matrix from factors and interactions

Description

Converts all factors and interactions to indicator variables, suitable for passing to WinBUGS.

Usage

getDesignMatrix(formula, data, effects = NULL)
**Arguments**

- **formula** A formula object specifying the fixed effects for the model
- **data** A data frame containing the covariates and factors for random effects
- **effects** A vector of character strings containing the grouping levels, from most general to most specific

**Details**

The most populous level of a factor is made the baseline.

**Value**

A matrix containing the covariates, the response(s), and the random effect factors. Also attributes

- **covariates** A list giving the covariates which apply at each level, suitable for passing to `winBugsRaggedArray`
- **response** A vector of character strings giving the responses

**Author(s)**

Patrick Brown

**See Also**

`winBugsRaggedArray`, `glmmBUGS`

**Examples**

```r
library(nlme)
data(Muscle)
muscleDesign = getDesignMatrix(conc ~ length, data=Muscle, effects="Strip")
attributes(muscleDesign)$covariates
attributes(muscleDesign)$response
```

---

**getRaggedSeq**

*Get one sequence for a ragged array*

**Description**

This function is called by `winBugsRaggedArray`

**Usage**

```r
getRaggedSeq(data)
```

**Arguments**

- **data** a data frame with two columns
Value
The ragged sequence

Author(s)
Patrick Brown <patrick.brown@utoronto.ca>

See Also
winBugsRaggedArray

---

class = 'getStartingValues'  
Description
Extract starting values for an MCMC chain from glmmPQL results

Usage
getStartingValues(pql, ragged, prefix=NULL, reparam=NULL)

Arguments
- pql: output from the glmmPQLstrings function
- ragged: a ragged array, from winBugsRaggedArray
- prefix: string to append to object names
- reparam: vector of random effect names, subtract covariates at this level from the intercept.

Details
This function produces a list suitable for passing to startingFunction to generate random starting values for use with bugs. If ragged has a spatial component, starting values for a spatial random effect will also be computed.

Value
A list of vectors, one for each set of parameters or random effects, and a list of estimated standard deviations.

Author(s)
Patrick Brown <patrick.brown@utoronto.ca>

See Also
glmmPQLstrings, startingFunction, bugs, glmmBUGS
Description

Creates ragged arrays, writes a model file, and generates sensible starting estimates.

Usage

```r
glmmBUGS(formula, data, effects, modelFile = "model.txt",
         initFile = "getInits.R",
         family = c("bernoulli", "binomial", "poisson", "gaussian"),
         spatial=NULL, spatialEffect = NULL,
         reparam=NULL, prefix=NULL, priors=NULL,
         brugs=length(grep("unix|linux",
                        .Platform$OS.type,
                        ignore.case=TRUE)))
```

Arguments

- `formula`: A formula for the fixed effects portion of the model
- `data`: A data frame containing the response, covariates, and group membership
- `effects`: A vector of character strings containing the grouping levels, from most general to most specific
- `modelFile`: File for saving the bugs model
- `initFile`: File for saving the function for generating initial values
- `family`: Distribution of responses
- `spatial`: For Markov Random Field models, a polygons or adjacency matrix. For Geostatistical models, a SpatialPoints objects, a matrix or data frame with columns "x" and "y", or a vector of complex numbers.
- `spatialEffect`: Spatial variable from data
- `reparam`: Vector of random effect names, subtract covariates at this level from the intercept.
- `prefix`: String to append to object names
- `priors`: List or vector where names refer to parameters and elements are prior distributions, for example `list(SDsite="dunif(0,10)").`
- `brugs`: Compatibility with OpenBUGS, using the inprod function in place of inprod2, defaults to FALSE on windows and TRUE on unix platforms.
Details

Consider the following model, where $Y_{ijk}$ is the number of absences from individual $k$ from class $j$ in school $k$.

$$Y_{ijk} \sim \text{Poisson}(\mu_i)$$

$$\log(\mu_i) = \delta age_{ijk} + \text{classSize}_ij\alpha + \text{schoolCategory}_i\gamma + U_i + V_{ij}$$

$$U_i \sim N(0, \sigma^2)$$

$$V_{ij} \sim N(0, \nu^2)$$

Here there are covariates which apply to each of the three levels, and random effects at the school and class level. If data is a data frame with one line per individual, the following would implement this model:

```r
glmmBUGS(data, effects=c("school","class"), covariates = list(school="schoolCategory", class="classSize"))
```

To aid in convergence, the bugs model is actually the following:

$$\log(\mu_i) = age_{ijk}\beta + V_{ij}$$

$$V_{ij} \sim N(U_i + \text{classSize}_ij\alpha, \nu^2)$$

$$U_i \sim N(\delta + \text{schoolCategory}_i\gamma, \sigma^2)$$

and the function `restoreParams` subtracts the means from the random effects to restore the original set of equations.

`glmmBUGS` calls the following functions:

- `getDesignMatrix` to convert factors and interactions to indicator variables and find which covariates apply at which levels
- `winbugsraggedarray` to prepare the ragged array
- `glmmPQLstrings` to estimate starting values
- `writebugsmodel` to create a model file
- `getStartingValues` to extract starting values from the glmmPQL result
- `startingFunction` to write a function to generate random starting values

Type `glmmBUGS` on the R command line to see the source code, it provides a good summary of the roles of the various functions in the `glmmBUGS` package.

Value

Returns a list with the ragged array, from `winbugsraggedarray`, and the list of starting values from `getStartingValues`. Writes a model file and an initial value function. Note that the initial value function in `initfile` will look for an object called `startingValues`, which does not exist as this is part of a list. Either create `startingValues <- result$startingValues` or edit `initfile`. 
Warning

You are strongly encouraged to modify the model file and the initial value function file prior to using them.

Note

glmmBUGS uses the inprod2 function, which isn’t implemented in OpenBugs, the model file will have to be modified for use with OpenBUGS.

Author(s)

Patrick Brown, <patrick.brown@utoronto.ca>

References


See Also

winbugsRaggedArray, glmmpqlstrings, writeBugsModel, getStartingValues, startingFunction.bugs

Examples

library(nlme)
data(Muscle)

muscleRagged = glmmBUGS(conc ~ length, data=Muscle, effects="Strip", family="gaussian")
startingValues = muscleRagged$startingValues

## Not run:
# run with winbugs
source("getInits.R")
require(R2WinBUGS)
muscleResult = bugs(muscleRagged$ragged, getInits, parameters.to.save = names(getInits()),
model.file="model.bug", n.chain=3, n.iter=1000, n.burnin=100, n.thin=10,
program="winbugs", working.directory=getwd())

# a jags example
require(R2jags)
muscleResultJags = jags(
muscleRagged$ragged, getInits, parameters.to.save = names(getInits()),
model.file="model.bug", n.chain=3, n.iter=1000,
 n.burnin=100, n.thin=10,
 working.directory=getwd())

## End(Not run)
data(muscleResult)

muscleParams = restoreParams(muscleResult, muscleRagged$ragged)

summaryChain(muscleParams)

checkChain(muscleParams)

# a spatial example
## Not run:
library(diseasemapping)

data(popdata)
data(casedata)

model = getRates(casedata, popdata, ~age*sex)
ontario = getSMR(popdata, model, casedata)
ontario = ontario@data[,c("CSDUID","observed","logExpected")]

library(spdep)
popDataAdjMat = poly2nb(popdata, row.names = as.character(popdata[["CSDUID"]]))

data(popDataAdjMat)
data(ontario)

forBugs = glmmBUGS(formula=observed + logExpected ~ 1,
  effects="CSDUID", family="poisson", spatial=popDataAdjMat,
  spatialEffect="CSDUID",
  data=ontario)

startingValues = forBugs$startingValues

source("getInits.R")

library(R2WinBUGS)
ontarioResult = bugs(forBugs$ragged, getInits,
  parameters.to.save = names(getInits()),
  model.file="model.bug", n.chain=3, n.iter=50, n.burnin=10, n.thin=2,
  program="winbugs", debug=T,working.directory=getwd())

data(ontarioResult)

ontarioParams = restoreParams(ontarioResult, forBugs$ragged)

ontarioSummary = summaryChain(ontarioParams)

# posterior probability of having 10x excess risk
postProb = apply(ontarioParams$FittedRCSDUID, 3, function(x) mean(x>log(10)) )

hist(postProb)
ontario = mergeBugsData(popdata, ontarioSummary)

spplot(ontario, "FittedRateCSDUID.mean")

ontario = mergeBugsData(ontario, postProb, newcol="postProb", by.x="CSDUID")
spplot(ontario, "postProb")

## End(Not run)

# geostatistical example

## Not run:
library(geoRglm)
data(rongelap)

rongelap = read.table(url(

library(spdep)
coordinates(rongelap) = ~X+Y

rongelap$logOffset = log(rongelap$time)
rongelap$site = seq(1, length(rongelap$time))

forBugs = glmmBUGS(
  formula=counts + logOffset ~ 1, family="poisson",
  data=rongelap$data, effects="site", spatial=rongelap,
  priors=list(phisite=dgamma(100,1)))

startingValues = forBugs$startingValues
startingValues$phi$site = 100

source("getInits.R")

rongelapResult = bugs(forBugs$ragged, getInits,
  parameters.to.save = names(getInits()),
  model.file="model.bug", n.chain=2, n.iter=20, n.burnin=4, n.thin=2,
  program="winbugs", debug=TRUE,
  working.directory=getwd())

data(rongelapResult)
glmmPQLstrings  

**An alternative interface to glmmPQL**

**Description**

Calls glmmPQL in the MASS library, with the model being specified in the same manner as `writeBugsModel`

**Usage**

```r
glmmPQLstrings(effects, covariates, observations, data = NULL, family = c("bernoulli", "binomial", "poisson", "gaussian"), ...)
```

**Arguments**

- `effects`: A vector of character strings containing the grouping levels, from most general to most specific
- `covariates`: A list with names corresponding to effects and each element being a vector of covariates applicable at that level
- `observations`: A character string giving the column of observations, or a vector where the first element is the observations and the remaining are offsets. For binomial responses, the first element is the counts (of successes), and the second element is the total number of trials. Note this differs from glmmPQL and glm’s notation, but is consistent with WinBUGS.
- `data`: A data frame containing the response, covariates, and group membership.
- `family`: The distribution to use. Either using `glmmPQL`’s specifications or `writeBugsModel`
- `...`: Further arguments to `glmmPQL`

**Details**

This function is useful for generating starting values for an MCMC chain.
Value

In addition to the output from glmmPQL, the following are returned:

effects, covariates, observations

As input

Author(s)

Patrick Brown, patrick.brown@utoronto.ca

See Also

glmmPQL

Examples

```r
library(nlme)
data(Muscle)
glmmPQL(strings(effects="Strip", observations="conc",
covariates=list(observations="length"),
data=Muscle, family="gaussian")
```

---

**muscleResult**  
*data set contains muscle result*

Description

Results from running the muscle example in `glmmBUGS`.

Usage

```r
data(muscleResult)
```

Format

A list as returned by the `bugs` function.

Details

See `glmmBUGS` and `Muscle`.

Examples

```r
data(muscleResult)
```
ontario

Description
Data frame showing expected and observed counts of molar cancer in Ontario

Usage
data(ontario)

Format
A data frame with 585 observations on the following 3 variables.

- CSDUID factor of Ontario census subdivision ID numbers
- observed Observed molar cancer cases
- logExpected expected cases

Details
See the documentation for glmmBUGS for how this was created.

Examples
data(ontario)
head(ontario)

---

ontarioResult

Description
Results from running Winbugs on the ontario data

Usage
data(ontarioResult)

Format
A list, as produced by the bugs function.
**popDataAdjMat**

**Examples**

```r
data(ontarioResult)
ontarioParams = restoreParams(ontarioResult)
ontarioSummary = summaryChain(ontarioParams)
```

---

**Description**

The popDataAdjMat Data set contains the adjacency matrix which calculated from the poly2nb function.

**Usage**

```r
data(popDataAdjMat)
```

**Details**

It is a adjacency matrix denoting the neighbours of Ontario census subdivisions. Created by:

```r
library(diseasemapping); data(popdata); popDataAdjMat = poly2nb(ontario,row.names=as.character(ontario)).
```

**Examples**

```r
data(popDataAdjMat)
```

---

**restoreParams**

**Reparametrise bugs output**

**Description**

Undoes the parametrisation used in `writeBugsModel`, and gives the original names to random effect levels.

**Usage**

```r
restoreParams(bugsResult, ragged = NULL, extraX=NULL)
```
Arguments

bugsResult  Output from bugs, using a ragged array generated by `winBugsRaggedArray` and a model generated by `writeBugsModel`
ragged  The ragged array used to call bugs
extraX  Possible extra covariates for spatial regions with no data but do have predicted spatial effects.

Value

A list where each element is a matrix or an array. The first dimension is the number of realisations, the second the number of chains, and for vector-valued parameters and random effects, the third dimension is the length of the parameter.

If the model contains a spatial component, the result will have list entries the following:

`Reffect`  The random effect. In the case of spatial models this is the sum of the spatial and non-spatial random effects U+V.

`,`

`ReffectSpatial`  The spatial random effect for each region, if any

`FittedReffect`  The predicted values on the link scale, being the random effect plus intercept and effect of covariates.

Note

For spatial models, one fitted rate is computed for each region in the adjacency matrix, even though some of these regions may not have spatial or non-spatial random effects simulated in the bugs model. If a spatial random effect is missing (as happens with islands), a zero is added. If a non-spatial random effect is missing (as happens when a region does not have data), a value is simulated unconditionally from each iteration’s intercept and standard deviation for that effect. Note that this does not add on the effect of possible covariates for that region. This can be added via the extraX argument.

Author(s)

Patrick Brown patrick.brown@utoronto.ca

See Also

bugs
**rongelapResult** | *Rongelap Winbugs Results*
---

**Description**
Results from running Winbugs on the Rongelap data

**Usage**
```r
data(rongelapResult)
```

**Format**
A list, as produced by the bugs function.

**See Also**
glmmBUGS

**Examples**
```r
data(rongelapResult)
rongelapParams = restoreParams(rongelapResult)
```

---

**rongelapUTM** | *Rongelap data*
---

**Description**
A SpatialPointsDataFrame containing the Rongelap data, in a UTM projection.

**Usage**
```r
data(rongelapUTM)
```

**Details**
These coordinates were obtained by translating and rotating the original Rongelap data until all the coordinates fit into the Rongelap border given by www.gadm.org. So they are not exact.

**Source**
See the help file for rongelap in geoRglm, or http://www.leg.ufpr.br/doku.php/pessoais:paulojus:mbgbook:datasets
startingFunction

Examples

data(rongelapUTM)
## Not run:
library(rgdal)
rongelapLL<spTransform(rongelapUTM, CRS("+proj=longlat +datum=NAD83"))
load(url("http://www.gadm.org/data/rda/MHL_adm8.RData"))
plot(rongelapLL)
plot(gadm, add=T)

## End(Not run)

startingFunction  Write a function to generate random MCMC starting values

Description

The code for the resulting function is saved in a file, to be edited and sourced in before calling WinBUGS.

Usage

startingFunction(startingValues, file = "getInits.R")

Arguments

startingValues  list returned from getStartingValues
file  character string giving the name of the file to write to

Details

Given a list containing initial estimates of parameters and random effects, a text file is produced containing code for a function to generate random starting values for use with the bugs() function. It is intended that the file produced be checked and edited prior to use.

Value

A file, with the name given by the 'file' argument, is written.

Warning

You are strongly encouraged to edit the file to ensure the result is sensible

Author(s)

Patrick Brown, patrick.brown@utoronto.ca

See Also

getStartingValues, bugs
summaryChain

Examples

```r
# Should be DIRECTLY executable !! ----
# Define data, use random,
# or do help(data=index) for the standard data sets.
sval = list(intercept=0, beta = 1:2, Rperson = rep(0, 5), vars=list(person=1))
startingFunction(sval)
```

summaryChain

*Compute mean, standard deviation, and quantiles of an MCMC run*

Description

Computes summary statistics for each parameter.

Usage

```r
summaryChain(chain, probs = c(0.005, 0.025, 0.05, 0.5))
```

Arguments

- `chain` The result from `restoreParams`, or the `sims.array` component of a `bugs` call.
- `probs` Quantiles for the posterior credible interval

Value

A list of matrices, with rows corresponding to summary statistics and columns to parameters.

- `scalar` Matrix for the scalar parameters
- `...` One matrix for each vector valued parameter
- `FittedRateEffect` For spatial models only, summaries on the natural scale (exponential of FittedReffect).

Author(s)

Patrick Brown

See Also

- `restoreParams`

Examples

```r
# create a simple chain
thechain = list(beta = array(1, c(10, 3, 4)),
               dimnames = list(NULL, NULL, paste("beta\%", 1:4, "\%", sep=""))),
               intercept = matrix(1, 10, 3))

summaryChain(thechain)
```
Ragged Arrays for multilevel models in BUGS

Description

Suitable for unbalanced data.

Usage

```r
winbugsraggedarray(data, effects = names(data)[-length(names(data))],
                    covariates = NULL, observations = names(data)[length(names(data))],
                    returnData = FALSE,
                    prefix=NULL, reparm=FALSE)
```

Arguments

- `data`: A data frame containing the response, covariates, and group membership.
- `effects`: A vector of character strings containing the grouping levels, from most general to most specific. Defaults to the column names of data, excluding the last column.
- `covariates`: A list with names corresponding to effects and each element being a vector of covariates applicable at that level.
- `observations`: A character string giving the column of observations, or a vector where the first element is the observations and the remaining are offsets.
- `returnData`: If true, returns the re-ordered data frame as well as the data frame.
- `prefix`: Character string to be appended to variable names.
- `reparam`: Vector of effect names, reparametrize the intercept by subtracting the mean of covariates at this level.

Details

This function creates a list of data suitable for passing to the `bugs` function, suitable for implementation as a ragged array. The output can be passed to `getStartingValues` to manipulate the output from `glmPQLstrings`, and to `restoreParams` to restore the original parametrisation from `bugs` output.

Value

A list with the following components:

- `Nxx`: The number of levels in the most general grouping.
- `Syy`: Indexing sequences, one for each level. If `yy` is level `n`, level `n+1` has elements `Syy[1]` to `Syy[2]-1` belonging to the first category of level `n`.
- `Xyy`: Matrix or vector of covariates belonging to level `yy` vector of observations.
**writeBugsModel**

**Author(s)**

Patrick Brown, <patrick.brown@utoronto.ca>

**References**


**See Also**

`bugs`

**Examples**

```r
library(nlme)
data(Muscle)
muscleRagged = winBugsRaggedArray(Muscle, effects="Strip",
observations="conc",
covariates=list(observations="length"))
```

---

**writeBugsModel**  
*Write a bugs model file for a Generalised Linear Mixed Model*

**Description**

Given a list of effect groups, and the covariates associated with each level, a bugs model file is written using ragged arrays corresponding to output from `winBugsRaggedArray`.

**Usage**

```r
writeBugsModel(file, effects, covariates, observations, family = c("bernoulli", "binomial", "poisson", "normal", "other"), spatial = NULL, geostat=FALSE, prefix = "", reparam=NULL, brugs=TRUE, priors=NULL)
```

**Arguments**

- `file`  
a character string denoting the name of the bugs model file written.
- `effects`  
vector of effect groups
- `covariates`  
A list with names corresponding to effects and each element being a vector of covariates applicable at that level
- `observations`  
A character string giving the column of observations, or a vector where the first element is the observations and the remaining are offsets.
- `family`  
Response distribution
- `spatial`  
name of the spatial random effect
writeBugsModel

geostat  Is this a geostatistical random effect? Defaults to FALSE for the Besag, York and Mollie discrete spatial variation model
prefix  the prefix
reparam  vector of random effect names, subtract covariates at this level from the intercept.
brugs  make the model file compatible with OpenBugs by using the inprod function in place of inprod2
priors  character string of prior distributions, with the name of each element referring to the parameter it is the prior for

Details

The arguments to the function specify a generalised linear mixed model. A file containing code for a corresponding bugs model is written. The model uses ragged arrays to specify grouping factors, and includes covariates at the appropriate levels to aid in chain convergence. It is intended that the user will edit this file before it's use. The prior distributions in particular may not be appropriate.

Value

A file, suitable for passing to the bugs function in R2WinBUGS.

Warning

You are strongly encouraged to modify the model file prior to using it.

Author(s)

Patrick Brown, <patrick.brown@utoronto.ca>

References


Examples

writeBugsModel("model.bug", effects="Strip", observations="conc", covariates=list(observations="length"), family="normal", priors=c(intercept="dunif(-10,10)"))
Index

*Topic datasets
    muscleResult, 15
    ontario, 16
    ontarioResult, 16
    popDataAdjMat, 17
    rongelapResult, 19
    rongelapUTM, 19

addSpatial, 2

binToBinom, 3
bugs, 5, 8, 11, 15, 16, 18–24

checkChain, 4
cholInvArray, 5

getDesignMatrix, 6, 10
getRaggedSeq, 7
getStartingValues, 8, 10, 11, 15, 20, 22
glmmBUGS, 2, 7, 8, 9, 15, 16, 19
glmmPQL, 14, 15
glmmPQLstrings, 8, 10, 11, 14, 22

Muscle, 15
muscleResult, 15

ontario, 16
ontarioResult, 16

popDataAdjMat, 17

restoreParams, 5, 10, 17, 21, 22
rongelapResult, 19
rongelapUTM, 19

startingFunction, 8, 10, 11, 20
summaryChain, 5, 21

winBugsRaggedArray, 7, 8, 10, 11, 18, 22, 23
writeBugsModel, 10, 11, 14, 17, 18, 23