Package 'JAGStree'

January 20, 2025

Type Package

Title Automatically Write 'JAGS' Code for Hierarchical Bayesian Models on Trees

Version 1.0.1

Maintainer Mallory J Flynn <mallory.flynn@stat.ubc.ca>

Description
When relationships between sources of data can be represented by a tree, the generation of appropriate Markov Chain Monte Carlo modeling code to be used with 'JAGS' to run a Bayesian hierarchical model can be automatically generated by this package. Any admissible tree-structured data can be used, under the assumption that node counts are multinomial and branching probabilities are Dirichlet among sibling groups. The methodological basis used to create this package can be found in Flynn (2023) <http://hdl.handle.net/2429/86174>.

License GPL (>= 2)

URL https://github.com/malfly/JAGStree

Depends R (>= 4.3.0)

Imports data.tree, gtools, mcmcplots, stats, R2jags, tidyverse, DiagrammeR, AutoWMM

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Mallory J Flynn [cre, aut]

Repository CRAN

Date/Publication 2024-11-11 12:00:13 UTC

Contents

`data1` *Simple Tree Data 1*

Description

Small, artificially generated toy data set to demonstrate package functionality

Usage

```
data(data1)
```

Format

An object of class `"data.frame"`

from A node label and started point of directed edge (parent node)

to A node label and endpoint of directed edge (child node)

Estimate A numerical value assumed to be survey count belonging to 'to' node (integer)

Total A numerical value assumed to be survey sample size (integer)

Count A numerical value for marginal count if leaf node (integer)

Population A boolean value for if survey size is entire population (logical)

Description A string describing 'to' node (string)

References

This data set was artificially created for the JAGStree package.

Examples

```
data(data1)
head(data1)
```

data2 *Simple Tree Data 2*

Description

Small, artificially generated toy data set to demonstrate package functionality with a simpler data frame

Usage

```
data(data2)
```

Format

An object of class `"data.frame"`

from A node label and started point of directed edge (parent node)

to A node label and endpoint of directed edge (child node)

References

This data set was artificially created for the JAGStree package.

Examples

```
data(data2)
head(data2)
```

data3 *Simple Tree Data 3*

Description

Small, artificially generated toy data set to demonstrate package functionality with a multinomial distribution among node branching

Usage

```
data(data3)
```

Format

An object of class "data.frame"

from A node label and started point of directed edge (parent node)

to A node label and endpoint of directed edge (child node)

Estimate A numerical value assumed to be survey count belonging to 'to' node (integer)

Total A numerical value assumed to be survey sample size (integer)

Count A numerical value for marginal count if leaf node (integer)

Population A boolean value for if survey size is entire population (logical)

Description A string describing 'to' node (string)

References

This data set was artificially created for the JAGStree package.

Examples

```
data(data3)
head(data3)
```

---

makeJAGStree *makeJAGStree*

Description

Generates a .mod or .txt file with 'JAGS' code for Bayesian hierarchical model on tree structured data

Usage

```
makeJAGStree(data, prior = "lognormal", filename = "JAGSmodel.mod")
```

Arguments

data	A dataframe object representing tree-structure
prior	A string representing the choice of prior for the root node population size; can be set to "lognormal" (default) or "uniform"
filename	A string containing the file name for resulting output 'JAGS' model file; must end in .mod or .txt

Value

A .mod or .txt file that contains code ready to run a Bayesian hierarchical model in 'JAGS' based on the input tree-structured data

Examples

```
# optional use of the AutoWMM package to show tree structure
Sys.setenv("RGL_USE_NULL" = TRUE)
tree <- makeTree(data1)
drawTree(tree)

makeJAGStree(data1, filename=file.path(tempdir(), "data1_JAGSscript.mod"))
makeJAGStree(data1, filename=file.path(tempdir(), "data1_JAGSscript.txt"))

# second example
makeJAGStree(data2, filename=file.path(tempdir(), "data2_JAGSscript.mod"))
makeJAGStree(data2, filename=file.path(tempdir(), "data2_JAGSscript.mod", prior = "uniform"))

# third example, showing optional execution with MCMC in R
makeJAGStree(data3, filename=file.path(tempdir(), "multiScript.mod"))
makeJAGStree(data3, filename=file.path(tempdir(), "multiScript.txt"))

mcmc.data <- list( "DE" = c(50, NA),
"ABC" = c(NA, 500, NA),
"pZ1" = 4,     # dirichlet parameters for prior on u
"pZ2" = 5,      # last parameter to sample U
"pZ3" = 1,
"pA1" = 10,      # beta parameters for prior on p
"pA2" = 1,
"mu" = log(1000),    # lognormal mean Z
"tau" = 1/(0.1^2))    # lognormal precision (1/variance) of Z

## define parameters whose posteriors we are interested in
mod.params <- c("Z",  "ABC", "DE", "pZ", "pA")

## modify initial values
mod.inits.cont <- function(){ list("Z.cont" = runif(1, 700, 1500),
                                   "ABC" = c(round(runif(1, 100, 200)), NA, NA),
                                   "pA" = as.vector(rbeta(1,1,1)),
                                   "pZ" = as.vector(rdirichlet(1, c(1,1,1))))
                                 }

## Generate list of initial values to match number of chains
numchains <- 6
mod.initial.cont <- list()
i <- 1
while(i <= numchains){
mod.initial.cont[[i]] <- mod.inits.cont()
i = i+1
}

## now fit the model in JAGS
mod.fit <- jags(data = mcmc.data,
              inits = mod.initial.cont,
              parameters.to.save = mod.params,
              n.chains = numchains,
              n.iter = 500, n.burnin = 200,
```

```
              model.file = "multiScript.mod")
print(mod.fit)

## plots using mcmcplots library
mod.fit.mcmc <- as.mcmc(mod.fit)
denplot(mod.fit.mcmc, parms = c("Z", "ABC[1]", "ABC[3]"))
denplot(mod.fit.mcmc, parms = c("pZ", "pA"))
traplot(mod.fit.mcmc, parms = c("Z", "ABC[1]", "ABC[3]"))
traplot(mod.fit.mcmc, parms = c("pZ", "pA"))
```

Index