

# Package ‘bayesmix’

June 14, 2009

**Version** 0.6-1

**Date** 2009-06-12

**Author** Bettina Gruen

**Maintainer** Bettina Gruen <Bettina.Gruen@wu.ac.at>

**Title** Bayesian Mixture Models with JAGS

**Depends** R (>= 2.6), coda

**SystemRequirements** JAGS 1.0.3

**Description** Bayesian mixture models of univariate Gaussian distributions using JAGS

**URL** <http://statmath.wu-wien.ac.at/~gruen/BayesMix>

**License** GPL-2

**Repository** CRAN

**Date/Publication** 2009-06-14 19:07:38

## R topics documented:

BMMdiag . . . . .	2
BMMmodel . . . . .	3
BMMposteriori . . . . .	4
BMMpriors . . . . .	5
darwin . . . . .	6
fish . . . . .	7
haveJAGS . . . . .	7
initsFS . . . . .	8
JAGScall . . . . .	9
JAGScontrol . . . . .	9
JAGSread . . . . .	10
JAGSrun . . . . .	11
JAGSsetup . . . . .	13

plot.BMMposteriori . . . . .	14
plot.jags . . . . .	15
priors . . . . .	16
randomPermutation . . . . .	16
Sort . . . . .	17

<b>Index</b>	<b>18</b>
--------------	-----------

---

BMMdiag	<i>Plot Identifiability Diagnostics for jags Object</i>
---------	---

---

## Description

Two different plots are currently provided: a plot of different variables against each other and a plot of the same variable against its values in the other classes.

## Usage

```
BMMdiag(object, which = 1:2, variables, ask = interactive(), fct1,
         fct2, xlim, ylim, auto.layout = TRUE, caption = NULL,
         main = "", ...)
```

## Arguments

object	a jags object with model of class BMMmodel.
which	if only one of the plots is required, specify its number.
variables	if variables is missing, the names are taken from the jags object.
ask	prompt user before each page of plots
fct1	string: name of transformation function for variable on x-axis.
fct2	string: name of transformation function for variable on y-axis.
xlim	if no range for xlim is specified, a sensible range is taken.
ylim	if no range for ylim is specified, a sensible range is taken.
auto.layout	logical: if TRUE puts each of the two different plots on one figure.
caption	captions to appear above the plots.
main	title to each plot (in addition to the above 'caption').
...	further graphical parameters (see 'plot.xy' and 'par') may also be supplied as arguments.

## Details

The plots help determining which variable will induce a unique labelling when taken for ordering of the segments and indicate if the model is overfitted by specifying too many segments.

## Author(s)

Bettina Gruen

---

BMMmodel

*Creates text for .bug-file and data for -inits.R and -data.R-file*


---

**Description**

Creates the text for the BUGS-model specification and the values of the initialization, prior specification and the observations read in by jags.

**Usage**

```
BMMmodel(y, k, priors, inits = "initsFS", aprioriWeights = 1,
         no.empty.classes = FALSE, restrict = "none", ...)
```

**Arguments**

<code>y</code>	a numeric vector.
<code>k</code>	integer indicating the number of segments.
<code>priors</code>	specification of priors by a named list or a <code>BMMpriors</code> object.
<code>inits</code>	specification of initial values by a named list or string indicating the function to be called.
<code>aprioriWeights</code>	specification of prior of the a-priori weights. If <code>aprioriWeights</code> does not have length = <code>k</code> , there is an equal prior for the a-priori weights assumed.
<code>no.empty.classes</code>	logical: should it be prevented that empty classes arise during sampling.
<code>restrict</code>	one of "none", "mu", "tau".
<code>...</code>	further parameters for the function specified in <code>inits</code> .

**Details**

By default the function `initsFS` is called for generating initial values. Any other function specified by `inits` is assumed to have at least `x`, `k` and `restrict` as input parameters.

The parameter `restrict` indicates if a location-shift model ("tau"), a scale contaminated model ("mu") or a model where both variables vary over components shall be fitted.

If the logical `no.empty.classes` is `TRUE` there are observations added to the model that the classes are not empty. This signifies that the likelihood when sampling the class affiliations is changed thus that any data point which is sampled and is the last one in its class stays there.

**Value**

If `y` is specified there is an object of class `BMMmodel` returned with components:

<code>inits</code>	named list for -inits.R-file.
<code>data</code>	named list for -data.R-file.
<code>bugs</code>	text for .bug-file with prefix missing.

If `y` is missing there is an object of class `BMMsetup` returned containing the parameter specifications. When `JAGSsetup` is called with this object as model argument, `BMMmodel` is called with `y` and the other parameters as input arguments before creating the input files for `jags`.

### Author(s)

Bettina Gruen

### See Also

[JAGSrun](#), [initsFS](#)

### Examples

```
data(fish)
model <- BMMmodel(fish, k = 4, priors = list(kind = "independence",
      parameter = "priorsFish", hierarchical = "tau"),
      initialValues = list(S0 = 2))

model
```

---

BMMposteriori

*Plots aposteriori probabilities of data points*

---

### Description

Given a `jags` object with model of class `BMMmodel` the aposteriori probabilities are determined. If `plot=TRUE`, the resulting object of class `BMMposteriori` is plotted by calling `plot.BMMposteriori`.

### Usage

```
BMMposteriori(object, class, caption = NULL, plot = TRUE,
      auto.layout = TRUE, ...)
```

### Arguments

<code>object</code>	a <code>jags</code> object with model of class <code>BMMmodel</code> .
<code>class</code>	a vector of integers indicating for which classes the posterior probabilities shall be plotted. The default is all.
<code>caption</code>	captions to appear above the plots.
<code>plot</code>	logical indicating if a plot shall be made.
<code>auto.layout</code>	logical: if <code>TRUE</code> puts all classes in the posterior probabilities plot on the same figure.
<code>...</code>	further graphical parameters may also be supplied as arguments.

### Details

Given a `jags` object with model of class `BMMmodel` the a posteriori probabilities are calculated for the unique data points with respect to the components specified by `class`.

**Value**

There is a `BMMposteriori` object returned which is a list including the following components

<code>data</code>	vector of unique data points.
<code>post</code>	a matrix including the posteriori probability of the data points for each class.

**Author(s)**

Bettina Gruen

**See Also**

`plot.BMMposteriori`

---

`BMMpriors`

*Creates a 'BMMpriors' object*

---

**Description**

This function enables a comfortable creation of `BMMpriors` objects, which can be used for specifying the priors of a `BMMmodel`.

**Usage**

```
BMMpriors(specification, y, eps = 10^-16)
```

**Arguments**

<code>specification</code>	named list including <code>kind</code> , <code>parameter</code> , <code>hierarchical</code> and <code>mod</code> .
<code>y</code>	a numeric vector.
<code>eps</code>	a numeric value indicating the smallest value for flat priors.

**Details**

In `specification` `kind` can be used for specifying if an "independent" or a "conditionallyconjugate" prior shall be used. `parameter` can be a named list of values for the prior variables or a function name (e.g., "priorsUncertain", "priorsFish", "priorsRaftery"). `hierarchical` can be `NULL` or "tau" if a hierarchical prior shall be taken for  $\tau$ . `mod` is a named list which provides the possibility to override the values from `parameter`.

**Value**

There is an object `BMMpriors` returned with components

<code>name</code>	vector indicating which kind of prior is specified and if it is an hierarchical prior and if appropriate with respect to which variable.
<code>var</code>	list of variables in the prior and their specified values.

**Author(s)**

Bettina Gruen

**Examples**

```
data(fish)
priors <- BMMpriors(y = fish)
```

---

darwin

*Differences in heights between plants*

---

**Description**

A numeric vector containing 15 observations of differences in heights between pairs of self-fertilized and cross-fertilized plants grown in the same condition.

**Usage**

```
data(darwin)
```

**Format**

A numeric vector of length 15.

**Details**

Darwin's data set contains two extremely small values. Therefore, this data set can be used for outlier modelling.

**Source**

Abraham, B. and G. Box (1978) Linear models and spurious observations. *Applied Statistics*, **27**, 131–8.

**Examples**

```
data(darwin)
## Estimated sample density
plot(density(darwin[[1]]), ylim = c(0, 0.02), main = "Outlier modelling")
ss <- seq(-100, 100, by = 1)
## Normal density with estimated mean and sd of whole sample
lines(ss, dnorm(ss, mean = mean(darwin), sd = sd(darwin)), col = "red")
## Normal density with estimated mean and sd of sample, where the 2
## extremely small values are removed
lines(ss, dnorm(ss, mean = mean(darwin[-c(1:2),1]), sd = sd(darwin[-c(1:2),1])),
      col = "green")
```

---

fish	<i>Fish length data</i>
------	-------------------------

---

**Description**

A numeric vector containing 256 observations of fish lengths.

**Usage**

```
data(fish)
```

**Format**

A numeric vector of length 256.

**Details**

This data set can be used for modeling unobserved heterogeneity, as it can be assumed that underlying categories present in the data are the age groups to which the fish belong.

**Source**

D. M. Titterington, A. F. M. Smith and U.E. Makov (1985) *Statistical Analysis of Finite Mixture Distributions*. Wiley.

**Examples**

```
data(fish)
ss <- seq(-3, 13, by = 0.01)
hist(fish[[1]], 20, freq = FALSE, main = "Fish data")
lines(ss, dnorm(ss, mean(fish), sd(fish)), col = "red")
```

---

haveJAGS	<i>Checks the availability of jags</i>
----------	--

---

**Description**

Given the location of the jags executable, it is checked if it can be successfully called.

**Usage**

```
haveJAGS(jags = getOption("jags.exe"))
```

**Arguments**

jags                    string indicating location of jags executable. If NULL it is assumed that jags is in the search path.

**Value**

Returns a logical.

**Author(s)**

Bettina Gruen

---

initsFS                      *create initial values*

---

**Description**

Initial values for nodes are created after the suggestion in Sylvia Fruehwirth-Schnatter's book.

**Usage**

```
initsFS(x, k, restrict, initialValues = list())
```

**Arguments**

x	a numeric vector.
k	number of segments.
initialValues	additional initial values specifications.
restrict	one of "none", "mu", "tau".

**Details**

The initial values for  $\mu$  are determined by the quantiles of the data, those for  $\eta$  give equal weight on each segment and those for  $\tau$  are equal for all segments and estimated by the inverse of the IQR of the data divided by 1.34 and squared.

**Value**

A list with initial values for the parameter indicated by the name of the respective list element is returned.

**Author(s)**

Bettina Gruen

---

JAGScall	<i>Calls jags</i>
----------	-------------------

---

**Description**

Given the `prefix` of the `.cmd`-file and the path to `jags` the program `jags` is called in batch mode by inserting the `.cmd`-file.

**Usage**

```
JAGScall(prefix, jags, quiet = FALSE)
```

**Arguments**

<code>prefix</code>	string: name of <code>.cmd</code> -file.
<code>jags</code>	string: path to <code>jags</code> .
<code>quiet</code>	logical indicating whether error messages should be ignored.

**Details**

If `jags` is missing, it is assumed that the program `jags` is in the path such that it is called with `jags`.

**Value**

Returns the exit status.

**Author(s)**

Bettina Gruen

**See Also**

[JAGSrun](#)

---

JAGScontrol	<i>Creates text for .cmd file</i>
-------------	-----------------------------------

---

**Description**

Creates the text for the file where all the commands for `jags` are included.

**Usage**

```
JAGScontrol(variables, draw = 1000, burnIn = 0, seed, rng =  
  c("base::Wichmann-Hill", "base::Marsaglia-Multicarry",  
    "base::Super-Duper", "base::Mersenne-Twister"))
```

**Arguments**

<code>variables</code>	names of variables which shall be monitored.
<code>draw</code>	number of monitored draws.
<code>burnIn</code>	number of discarded burn-in draws.
<code>seed</code>	integer setting the seed for the RNG.
<code>rng</code>	specification of random number generator.

**Details**

This function creates the text for the `.cmd`-file. It includes information on the number of burn-in draws and monitored draws. Furthermore, it states which variables shall be monitored and it possibly specifies a seed. The information on the prefix of the files containing model, data and initial value is missing. By inserting the prefix between the elements of the vector containing the text a `.cmd`-file can be created which can be used for calling `jags` in batch mode.

**Value**

An object of class `JAGScontrol` is returned which is a list containing the following components:

<code>text</code>	a vector of strings which contains all commands for running <code>jags</code> . For creating a <code>.cmd</code> -file the prefix of the files has to be inserted.
<code>variables</code>	a vector containing the names of the monitored variables.

**Author(s)**

Bettina Gruen

**See Also**

[JAGSrun](#)

**Examples**

```
control <- JAGScontrol(variables = "mu")
control
```

---

JAGSread

*Reads jags output into R*

---

**Description**

Given an exit status of 0 when running `jags` the results are read in with `read.jags`

**Usage**

```
JAGSread(exit, transform = TRUE)
```

**Arguments**

exit                    exit status returned by JAGScall.  
transform            logical: if TRUE and possible, the variable  $\tau$  is transformed into  $\sigma^2$ .

**Details**

After a successful run of jags the results are read in from the file "jags.out". If the exit status is positive the file "jags.dump" is read in and a warning is given.

**Value**

A list with the following components is returned:

results                MCMC chains.  
variables            new variable names.

**Author(s)**

Bettina Gruen

**See Also**

read.jags, [JAGSrun](#)

---

JAGSrun

*MCMC sampling of Bayesian models*

---

**Description**

Calls jags for MCMC sampling.

**Usage**

```
JAGSrun(y, prefix = yname, model = BMMmodel(k = 2),
        control = JAGScontrol(variables = c("mu", "tau", "eta")), tmp = TRUE,
        cleanup = TRUE, jags = getOption("jags.exe"), ...)
```

**Arguments**

y                      a numeric vector.  
prefix                character prefix of files.  
model                object of class JAGSmodel or output from BMMmodel.  
control              specification of control by a JAGScontrol object.  
tmp                    logical: shall the files be written in a temporary directory.  
cleanup              logical: shall the created files be removed.

jags	string indicating location of jags executable.
yname	a character string with the actual y argument name.
...	further parameters handed over to <code>BMMmodel</code> where it is used for the function specifying the initial values, e.g., <code>initsFS</code> .

### Details

If an error occurs when running `jags`, the created files are not removed. This function is a wrapper calling `JAGSsetup`, `JAGScall` and `JAGSread`.

### Value

Returns a `jags` object with components

call	the matched call.
results	results read in from “jags.out” if run was successful or from “jags.dump” if an error occurred.
model	a <code>JAGSmodel</code> object.
variables	vector containing the names of the monitored variables.
data	a numeric vector.

### Author(s)

Bettina Gruen

### See Also

[JAGSsetup](#), [JAGScall](#), [JAGSread](#), [BMMmodel](#), [initsFS](#)

### Examples

```
data(fish)
prefix <- "fish"
variables <- c("mu", "tau", "eta")
k <- 3
modelFish <- BMMmodel(k = k, priors = list(kind = "independence",
      parameter = "priorsFish", hierarchical = "tau"))
controlFish <- JAGScontrol(variables = c(variables, "S"), draw = 100, seed = 1)
## Installation of JAGS necessary for applying these functions
if (haveJAGS()) {
z1 <- JAGSrun(fish, prefix, model = modelFish, initialValues = list(S0 = 2),
      control = controlFish, cleanup = TRUE, tmp = FALSE)
zSort <- Sort(z1, "mu")
BMMposteriori(zSort)
}
data(darwin)
prefix <- "darwin"
k <- 2
modelDarwin <- BMMmodel(k = k, priors = list(kind = "independence",
      parameter = "priorsUncertain"), aprioriWeights = c(1, 15),
```

```

                                no.empty.classes = TRUE, restrict = "tau")
## Installation of JAGS necessary for applying these functions
if (haveJAGS()) {
z2 <- JAGSrun(darwin, prefix, model = modelDarwin, control =
              JAGScontrol(variables = variables, draw = 3000, burnIn = 1000,
                           seed = 1), cleanup = TRUE, tmp = FALSE)
plot(z2, variables = "mu")
}

```

---

JAGSsetup

*Creation of necessary input files for jags*


---

## Description

Given the model and the control specifications the input files (.bug, -data.R, -inits.R, .cmd) for jags are created with `prefix` as their names.

## Usage

```

JAGSsetup(model, y, prefix, control, ...)

## Default S3 method:
JAGSsetup(model, y, prefix, control, ...)
## S3 method for class 'BMMsetup':
JAGSsetup(model, y, prefix, control, ...)

```

## Arguments

<code>model</code>	JAGSmodel object or output from BMMmodel.
<code>y</code>	a numeric vector.
<code>prefix</code>	string: prefix for .bug, -data.R, -inits.R and .cmd-file.
<code>control</code>	named list or JAGScontrol object.
<code>...</code>	additional parameters handed over to BMMmodel.

## Details

`control` needs to have a component `text` which is written into the .cmd file after inserting the `prefix` where necessary.

## Value

After creating the files `prefix"-data.R"`, `prefix"-inits.R"`, `prefix".bug"` and `prefix".cmd"` it returns a list with the following components

<code>control</code>	input argument <code>control</code> with <code>prefix</code> inserted in the <code>componenttext</code> .
<code>model</code>	JAGSmodel object

**Author(s)**

Bettina Gruen

**See Also**

[BMMmodel](#), [JAGScontrol](#)

---

`plot.BMMposteriori` *Plots a posteriori probabilities of data points*

---

**Description**

Plot method for object of class `BMMposteriori`, typically called by `BMMposteriori`.

**Usage**

```
## S3 method for class 'BMMposteriori':  
plot(x, caption, main = "", ...)
```

**Arguments**

<code>x</code>	a <code>BMMposteriori</code> object.
<code>caption</code>	captions to appear above the plots.
<code>main</code>	title to each plot-in addition to the above <code>caption</code> .
<code>...</code>	further graphical parameters may also be supplied as arguments.

**Details**

This function is called by `BMMposteriori` if `plot = TRUE`.

**Author(s)**

Bettina Gruen

**See Also**

[BMMposteriori](#)

---

plot.jags

*Plot jags Object*


---

## Description

Plots mcmc chains of a jags object.

## Usage

```
## S3 method for class 'jags':
plot(x, variables = NULL, trace = TRUE, density = TRUE,
     smooth = TRUE, bwf, num, xlim, auto.layout = TRUE, ask = interactive(), ...)
```

## Arguments

x	a jags object.
variables	names of variables which shall be plotted. Default are all names of results except those with a column dimension larger than the number of classes k.
trace	plot trace of each variable.
density	plot density estimate of each variable.
smooth	draw a smooth line through trace plots.
bwf	bandwidth function for density plots.
num	if not all classes of a variable shall be plotted, a subset can be specified.
xlim	if not specified, the range of each variable over all classes is taken as default.
auto.layout	automatically generate output format.
ask	prompt user before each page of plots.
...	further arguments for densityplot.

## Details

Adapted from `plot.mcmc`.

Currently only implemented for jags objects with model of class `BMMmodel`. Otherwise the default plot method for the results of the jags object is called (`plot.mcmc`).

## Author(s)

Bettina Gruen

## See Also

`plot.mcmc`, [BMMdiag](#), [BMMposteriori](#)

---

priors *Creates list of prior specifications*

---

### Description

Given the data values for the priors are determined.

### Usage

```
priorsFish(y, eps = 10^-16)
priorsRaftery(y)
priorsUncertain(y, eps = 10^-16)
```

### Arguments

y a numeric vector.  
eps a numeric value indicating the smallest value for flat priors.

### Details

Values for the prior parameter  $b_0$ ,  $B_0$ ,  $\nu_0$  and  $S_0$  are determined.

### Value

There is a list returned with named components of the prior parameters.

### Author(s)

Bettina Gruen

---

randomPermutation *Randomly permute segments for MCMC draws*

---

### Description

Random permutation of segment labels for each draw in order to get a better estimate of the unrestricted likelihood.

### Usage

```
randomPermutation(x)
```

### Arguments

x a jags object with model of class BMMmodel.

**Details**

The draws are permuted with respect to the different classes  $k$ .

**Value**

The input object with permuted results for each draw is returned.

**Warning**

Any variables where there are neither  $k$  different chains nor only one chain observed are dropped.

**Author(s)**

Bettina Gruen

---

Sort

*Sorts MCMC chains according to certain variables*

---

**Description**

Ascending sorting of results of `jags` object with model of class `BMMmodel` with respect to a given variable.

**Usage**

```
Sort(x, by = NULL)
```

**Arguments**

<code>x</code>	a <code>jags</code> object with model of class <code>BMMmodel</code> .
<code>by</code>	variable name according to which the segments shall be ordered.

**Details**

If `by` is not specified, the first variable in the corresponding vector of the `jags` object is taken.

**Value**

The input object with results sorted in ascending order according to the variable given in `by` is returned.

**Warning**

If there arise problems, the original object is returned with a warning.

**Author(s)**

Bettina Gruen

# Index

## \*Topic **datasets**

darwin, [6](#)  
fish, [7](#)

## \*Topic **hplot**

BMMdiag, [1](#)  
BMMposteriori, [4](#)  
plot.BMMposteriori, [14](#)  
plot.jags, [15](#)

## \*Topic **utilities**

BMMmodel, [2](#)  
BMMpriors, [5](#)  
haveJAGS, [7](#)  
initsFS, [8](#)  
JAGScall, [9](#)  
JAGScontrol, [9](#)  
JAGSread, [10](#)  
JAGSrun, [11](#)  
JAGSsetup, [13](#)  
priors, [16](#)  
randomPermutation, [16](#)  
Sort, [17](#)

plot.BMMposteriori, [5](#), [14](#)  
plot.jags, [15](#)  
print.jags (*JAGSrun*), [11](#)  
print.JAGScontrol (*JAGScontrol*), [9](#)  
print.JAGSmodel (*BMMmodel*), [2](#)  
priors, [16](#)  
priorsFish (*priors*), [16](#)  
priorsRaftery (*priors*), [16](#)  
priorsUncertain (*priors*), [16](#)  
  
randomPermutation, [16](#)  
  
Sort, [17](#)

BMMdiag, [1](#), [15](#)  
BMMmodel, [2](#), [12](#), [14](#)  
BMMposteriori, [4](#), [14](#), [15](#)  
BMMpriors, [5](#)

darwin, [6](#)

fish, [7](#)

haveJAGS, [7](#)

initsFS, [3](#), [8](#), [12](#)

jags (*JAGSrun*), [11](#)

JAGScall, [9](#), [12](#)

JAGScontrol, [9](#), [14](#)

JAGSread, [10](#), [12](#)

JAGSrun, [3](#), [9](#), [10](#), [11](#), [11](#)

JAGSsetup, [12](#), [13](#)