Package: bmhe (via r-universe)

November 17, 2024

Type Package

Title This Package Creates a Set of Functions Useful for Bayesian modelling

Version 0.1.0

Author Gianluca Baio

Description A set of utility functions that can be used to post-process BUGS or JAGS objects as well as other to facilitate various Bayesian modelling activities (including in HTA).

Depends ggplot2, dplyr, BCEA

Suggests R2OpenBUGS, R2jags, manipulate, purrr, tidyr

License CC BY 4.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1.9000

Config/pak/sysreqs make libicu-dev

Repository https://giabaio.r-universe.dev

RemoteUrl https://github.com/giabaio/bmhe_utils

RemoteRef HEAD

RemoteSha cf98bc33daad0730f32911991ac3d72bd680a297

Contents

acfplot *Autocorrelation plot*

Description

Plots the ACF function

Usage

```
acfplot(x, col = "black", parameter = NULL, add_deviance = TRUE, ...)
```

Arguments

`x`	A vector with simulations from a MCMC process (eg from a BUGS or JAGS run)
`col`	The color with which to plot the ACF (default to "black")
`parameter`	A text string to select a named parameter (eg if using a BUGS or JAGS object, that would be one of the monitored parameters)
`add_deviance`	a logical argument to determine whether the deviance should be added to the plot (in case it is monitored). Defaults to TRUE,but is only relevant if the input object x is a BUGS or JAGS object
`...`	Extra arguments

Author(s)

Gianluca Baio

Examples

```
## Not run:
acfplot(m)

## End(Not run)
```

`betaPar` *Computes the parameters of a Beta distribution so that the mean and standard dev are the input (m,s)*

Description

Computes the parameters of a Beta distribution so that the mean and standard dev are the input (m,s)

Usage

```
betaPar(m, s)
```

Arguments

m	The implied mean for the underlying Beta distribution
s	The implied standard deviation for the underlying Beta distribution

Value

The list of relevant output including the values for the parameters of the Beta distribution (alpha and beta)

Examples

```
betaPar(.5,.15)
```

`betaPar2` *Compute the parameters of a Beta distribution, given a prior guess for key parameters. Based on "Bayesian ideas and data analysis", page 100. Optimisation method to identify the values of a,b that give required conditions on the Beta distribution*

Description

Compute the parameters of a Beta distribution, given a prior guess for key parameters. Based on "Bayesian ideas and data analysis", page 100. Optimisation method to identify the values of a,b that give required conditions on the Beta distribution

Usage

```
betaPar2(mode, upp, prob)
```

Arguments

`mode`	The implied mode of the distribution
`upp`	An upper bound value for the distribution
`prob`	The estimated probability that theta <= upp

Value

The list of relevant output including the values for the parameters of the Beta distribution and some underlying summary statistics of the resulting variable

Examples

```
res=betaPar2(.6,.7,.9)
```

`betaplot` *Trial-and-error Beta plot*

Description

Provides a quick and dirty, trial-and-error tool to identify suitable values for the the parameters of a Beta distribution to match set properties (eg mean, sd, 95% interval)

Usage

```
betaplot(a_max = 30, b_max = 30, step = 0.01)
```

Arguments

`a_max`	The maximum value for the parameter `a` of the Beta distribution
`b_max`	The maximum value for the parameter `b` of the Beta distribution
`step`	The increment in the grid of values for `a` and `b`

Author(s)

Gianluca Baio

Examples

```
## Not run:
betaplot()

## End(Not run)
```

`coefplot` *Coefplot for the parameters in the model*

Description

Creates a plot showing the mean and an interval estimate for the posterior distributions in a given model.

Usage

```
coefplot(x, low = 0.025, upp = 0.975, parameter = NULL, ...)
```

Arguments

`x`	an object of class 'bugs', see `BUGS`, or of class 'jags', see `JAGS` for details
`low`	the lower quantile to consider (default 2.5 percentile)
`upp`	the upper quantile to consider (default 97.5 percentile)
`parameter`	a vector of strings with the names of the parameters to be included. Defaults to all those in the original model, but can be a vector eg `c("par1","par2")`
`...`	Additional options

Author(s)

Gianluca Baio

See Also

`BUGS`, `JAGS`

Examples

```
## Not run:
coefplot(m)

## End(Not run)
```

diagplot *Specialised diagnostic plots*

Description

Creates a plot showing the output of convergence indicators, such as the Potential Scale Reduction and the effective sample size

Usage

```
diagplot(x, what = "Rhat", label = FALSE, ...)
```

Arguments

`x`	an object of class ‘bugs’, see `BUGS`, or of class ’jags’, see `JAGS` for details
`what`	A string indicating what diagnostic measure should be plotted. Options are ’Rhat’ (default), indicating the PSR statistic, or ’n.eff’, indicating the effective sample size
`label`	A logical input. If set to ’FALSE’ (default), then does not include text labels next to each node
`...`	Additional options

Author(s)

Gianluca Baio

See Also

`BUGS`, `JAGS`

Examples

```
## Not run:
diagplot(m)

## End(Not run)
```

gammaPar *Computes the parameters of a Gamma distribution so that the mean and standard dev are the input (m,s)*

Description

Computes the parameters of a Gamma distribution so that the mean and standard dev are the input (m,s)

Usage

```
gammaPar(m, s)
```

Arguments

m	The implied mean for the underlying Beta distribution
s	The implied standard deviation for the underlying Beta distribution

Value

The list of relevant output including the values for the parameters of the Gamma distribution (shape and rate)

Examples

```
gammaPar(12,3)
```

gammaplot *Trial-and-error Gamma plot*

Description

Provides a quick and dirty, trial-and-error tool to identify suitable values for the the parameters of a Gamma distribution to match set properties (eg mean, sd, 95% interval)

Usage

```
gammaplot(shape_max = 30, rate_max = 30, step = 0.01)
```

Arguments

shape_max	The maximum value for the parameter shape of the Gamma distribution
rate_max	The maximum value for the parameter rate of the Gamma distribution
step	The increment in the grid of values for shape and rate

Author(s)

Gianluca Baio

Examples

```
## Not run:
gammaplot()

## End(Not run)
```

`ilogit` *Computes the inverse logit of a number between -infinity and +infinity*

Description

Computes the inverse logit of a number between -infinity and +infinity

Usage

```
ilogit(x)
```

Arguments

`x`	a real number

Value

inverse-logit(x) = exp(x)/(1+exp(x))

Examples

```
ilogit(2)
```

`logit` *Computes the logit of a number*

Description

Computes the logit of a number

Usage

```
logit(x)
```

Arguments

`x` a number between 0 and 1

Value

logit(x)=log(x/(1-x))

Examples

```
logit(.2)
```

`logitPar` *Computes the parameters of a Normal distribution* on the logit scale*, so that,* on the natural scale*, the range where most of the mass is included is between the input ’low’ and ’upp’*

Description

Computes the parameters of a Normal distribution *on the logit scale*, so that, *on the natural scale*, the range where most of the mass is included is between the input ’low’ and ’upp’

Usage

```
logitPar(low, upp)
```

Arguments

`low` The lower extreme of an implied range that is supposed to cover "most" of the mass under the natural scale of the distribution of the parameter (defined in the interval 0–1)

`upp` The upper extreme of an implied range that is supposed to cover "most" of the mass under the natural scale of the distribution of the parameter (defined in the interval 0–1)

Value

The list of relevant output including the values for the parameters of the normal distribution (mulogit and sigmalogit), **on the logit scale**

Examples

```
logitPar(0.04,0.12)
```

lognPar *Computes mean and variance of a logNormal distribution so that the parameters on the natural scale are mu and sigma*

Description

Computes mean and variance of a logNormal distribution so that the parameters on the natural scale are mu and sigma

Usage

```
lognPar(m, s)
```

Arguments

`m`	The implied mean for the underlying Beta distribution
`s`	The implied standard deviation for the underlying Beta distribution

Value

The list of relevant output including the values for the parameters of the logNormal distribution in terms of the mean on the log scale (mulog) and the sd on the log scale (sigmalog)

Examples

```
lognPar(3,.15)
```

mytraceplot *Makes a traceplot (eg to visualise MCMC simulations from multiple chains)*

Description

Makes a traceplot (eg to visualise MCMC simulations from multiple chains)

Usage

```
mytraceplot(node, model = m, title = "", lab = "")
```

Arguments

`node`	a *string* with the name of the node to be plotted, eg "theta" (in quotes)
`model`	the name of the object containing the MCMC simulations
`title`	the title of the graph (defaults to nothing)
`lab`	the label to write on the y-axis (defaults to nothing)

Value

the graph with the traceplot

Examples

```
## Not run:
mytraceplot(”x”,m)

## End(Not run)
```

---

odds2probs *Maps from odds to probabilities*

Description

Maps from odds to probabilities

Usage

```
odds2probs(odds)
```

Arguments

odds	the odds ratio *against* p: OR=(1-p)/p

Value

the value of the underlying probability, p

Examples

```
odds2probs(4)
```

OR *Computes the odds ratio between two probabilities*

Description

Computes the odds ratio between two probabilities

Usage

```
OR(p1, p2)
```

Arguments

`p1` a probability

`p2` another probability

Value

OR=(p1/(1-p1))/(p2/(1-p2))

Examples

```
OR(.5,.2)
```

`plotGR` *Produces a plot of the values of the Gelman Rubin stats to determine visually convergence (and see clearly which node has reached it)*

Description

Produces a plot of the values of the Gelman Rubin stats to determine visually convergence (and see clearly which node has reached it)

Usage

```
plotGR(m)
```

Arguments

`m` is an object in the class jags or bugs (the output of the MCMC run)

Value

the graph with the Gelmn Rubin statistics plot

Examples

```
## Not run:
plotGR(m)

## End(Not run)
```

posteriorplot *Various plots for the posteriors in a* bugs *or* jags *object*

Description

Various plots for the posteriors in a bugs or jags object

Usage

```
posteriorplot(x, parameter = NULL, plot = "density", add_deviance = FALSE, ...)
```

Arguments

x	an object of class ‘bugs’, see BUGS, or of class ’jags’, see JAGS for details
parameter	a string with the name of the parameter for which to show the density plot. Can be a vector, eg c("par1","par2")
plot	the type of plot (options are ’density’ (default) or ’bar’ for a binned barplot of the posterior) or ’hist’ for a histogram
add_deviance	a logical argument to determine whether the deviance should be added to the plot (in case it is monitored). Defaults to FALSE
...	further arguments

Author(s)

Gianluca Baio

See Also

BUGS, JAGS

print.bugs *Printing a bugs object*

Description

Printing a bugs object

Usage

```
## S3 method for class 'bugs'
print(x, digits = 3, intervals = c(0.025, 0.25, 0.5, 0.75, 0.975), ...)
```

Arguments

`x` an object of class ‘bugs’, see bugs for details

`digits` rounding for tabular output on the console (default is to round to 1 decimal place)

`intervals` the quantiles for the posterior distribution to be displayed in the summary statistics table

`...` further arguments to `print`

Author(s)

Gianluca Baio

See Also

`bugs`

---

stats — *Computes and prints summary statistics for a vector or matrix of simulated values*

Description

Computes and prints summary statistics for a vector or matrix of simulated values

Usage

```
stats(x, dim = 2, out = "table", ...)
```

Arguments

`x` A vector or a matrix containing simulations from, eg, BUGS

`dim` The dimension alongside which the summaries should be taken (by default is 2, which means the simulations are stored as a matrix, where the variables are columns)

`out` A string indicating whether the output of the summary should be formatted as a "normal" vector or matrix (default), or as a tibble. Acceptable values are `"table"` (default) or `"tibble"`

`...` Additional parameters that can be passed for the option `out="tibble"`; includes `digits` (the number of significant digits to print) and `na.rm` (a logical value to indicate whether to remove the missing values from the calculations of the summaries)

Value

A table with some specific summary statistics

Examples

```
x=rnorm(1000)
stats(x)
```

`stats2` *Computes and prints summary statistics for a vector or matrix of simulated values - tidyverse style*

Description

Computes and prints summary statistics for a vector or matrix of simulated values - tidyverse style

Usage

```
stats2(x, digits = 3, na.rm = TRUE)
```

Arguments

`x`	A vector or a matrix containing simulations from, eg, BUGS
`digits`	The number of significant digits shown (default = 3)
`na.rm`	A logical value (default TRUE) to indicate whether NA should be removed

---

`traceplot` *Tidyverse based function to do traceplots*

Description

Traceplot for a `bugs` or `jags` object

Usage

```
traceplot(x, parameter = NULL, ...)
```

Arguments

`x`	an object of class ‘bugs’, see `BUGS`, or of class ’jags’, see `JAGS` for details. It can also be a vector with simulations for a single variable
`parameter`	a string with the name of the parameter for which to show the traceplot. Can be a vector, eg `c("par1","par2")`
`...`	further arguments to `traceplot`

Author(s)

Gianluca Baio

See Also

BUGS, JAGS

Index