Package 'mcmcOutput'

June 4, 2020

Type Package

Title Functions to Store, Manipulate and Display Markov Chain Monte Carlo (MCMC) Output

Version 0.1.1

Date 2020-05-29

Depends R (>= 2.10)

Imports HDInterval, coda, MASS

URL https://github.com/mikemeredith/mcmcOutput

BugReports https://github.com/mikemeredith/mcmcOutput/issues

Description Implements a class ('mcmcOutput') for efficiently storing and handling Markov chain Monte Carlo (MCMC) output, intended as an aid for those writing customized MCMC samplers. A range of constructor methods are provided covering common output formats. Functions are provided to generate summary and diagnostic statistics and to display histograms or density plots of posterior distributions, for the entire output, or subsets of draws, nodes, or parameters.

License GPL (>= 3)

NeedsCompilation no

Author Mike Meredith [aut, cre], John Kruschke [ctb]

Maintainer Mike Meredith <mike@mmeredith.net>

Repository CRAN

Date/Publication 2020-06-04 09:10:02 UTC

R topics documented:

`bigCrosscorr` *Cross-correlations for MCMC output*

Description

When MCMC output has hundreds of monitored nodes, the full cross-correlation matrix produced by `cor` is of little use. `bigCrossCorr` extracts and reports only those values greater than a given threshold.

Usage

```
bigCrosscorr(x, big = 0.6, digits = 3)
```

Arguments

`x`	an object of any class with MCMC output that can be coerced to class `mcmcOutput`.
`big`	only values below -big or above +big will be returned
`digits`	the number of decimal places to return

Value

A data frame with 2 columns for the names of parameters and a 3rd column with the cross-correlation, sorted in order of decreasing absolute cross-correlation.

Author(s)

Mike Meredith

See Also

`crosscorrPlot`.

`crosscorrPlot` *Plot image of correlation matrix*

Description

Displays graphically the lower triangle of the correlation matrix among the columns of the input.

Usage

```
crosscorrPlot(x, params=NULL, col, addSpace=c(0,0), ...)
```

Arguments

`x`	An object with MCMC chain values, of any class that can be coerced to `mcmcOutput`.
`params`	An optional vector of column numbers or names; names are partially matched, so `params="alpha"` will match all of `alpha,alpha0,alpha[1],alphanew`; negative indices indicate parameters to exclude, positive and negative indices cannot be mixed.
`col`	The colours to use to code the correlations; default is a blue-yellow-red ramp; NA correlations appear as white boxes.
`addSpace`	A length-2 vector to add extra white space right and above the main display, eg, to provide extra space for long parameter names; units are the width of one box of the display. Unlike `mar`, this inserts space left of the legend and below the title.
`...`	Additional graphical parameters, see Details.

Details

The usual graphical parameters can be added in the ... argument. Note the following:

* `mar` : a vector of length 4 specifying the width of the margins below, left, above and right of the main plot; you will need to increase the margins to insert `xlab` or `ylab`; default `c(1,1,5,4)`; see the entry for `mar` at `par`.

* `cex.axis` : controls the size of the parameter names, default 1.2.

* `srt` : controls the rotation of the parameter names, 0 = horizontal, 90 = vertical, default 45.

* `offset` : controls the distance of the start of the parameter names from the corner of the box in units of box width, default 0.2.

* `tcl` : the length of the tick marks next to the parameter names in units of box widths, default 0.1.

* `lwd.ticks` : line width for the tick marks, default 1.

* `legendAsp` : aspect ratio for the legend, default 0.1.

Value

Returns the correlation matrix invisibly.

Author(s)

Mike Meredith

Examples

```
# Create a data frame of fake MCMC output:
mu0 <- rnorm(3000)         # normal, mean zero
mu10 <- rnorm(3000, rep(9:11, each=1000), 1) + mu0*0.5
  # approx normal, mean 10, correlated with mu0
fake <- data.frame(
  mu0 = mu0,
  mu10 = mu10,
  sigma=rlnorm(3000),        # non-negative, skewed
  prob = plogis(1-mu0),      # probability, central mode, neg. correlation with mu0
  prob0 = rbeta(3000, 1,2),  # probability, mode = 0
  N = rpois(3000, rep(c(24, 18, 18), each=1000)),
                             # large integers (no zeros), poor mixing
  n = rpois(3000, 2),        # small integers (some zeros)
  const1 = rep(1, 3000))     # all values = 1
str(fake)

tmp <- crosscorrPlot(fake)
round(tmp, 2)
crosscorrPlot(fake, main="Isn't this a really cool plot?")
crosscorrPlot(fake, main="A subset of parameters", params=c("mu", "prob", "N"))
crosscorrPlot(fake, main="Leave out 'sigma'", params=-3)
crosscorrPlot(fake, main="Just a few colours", col=c("blue","skyblue","pink","red"))
names(fake)[5] <- "A_parameter_with_a_very_long_name"
crosscorrPlot(fake, main="Is there room?")
crosscorrPlot(fake, main="With addSpace=c(2,0)", addSpace=c(2,0))
```

`densityFolded` *Folded kernel density estimation*

Description

Parameters are often constrained to be greater than zero (eg, standard deviation) or within the range (0, 1) (eg, probabilities), but the function `density` often returns non-zero densities outside these ranges. Simple truncation does not work, as the area under the curve is < 1. The function `densityFolded` attempts to identify these constraints and gives an appropriate density.

If `x` is a matrix, detection of constraints and selection of bandwidth is applied to the pooled values, but a separate density curve is fitted to each column of the matrix.

Usage

```
densityFolded(x, bw = "nrd0", adjust = 1, from=NA, to=NA, ...)
```

Arguments

x	a numeric vector or matrix from which the estimate is to be computed; missing values not allowed.
bw	the smoothing bandwidth to be used; see density for details.
adjust	the bandwidth used is actually adjust*bw.
from, to	the lower and upper ends of the grid at which the density is to be estimated; if NA, range will cover the values in x; ignored and replaced with 0 or 1 if a constraint is detected.
...	other arguments passed to density.

Value

Returns a list containing the following components:

x the n coordinates of the points where the density is estimated.

y a vector or matrix with the estimated density values.

bw the bandwidth used.

n the sample size after elimination of missing values.

call the call which produced the result.

data.name the deparsed name of the x argument.

If y is a vector, the output will have class density.

Author(s)

Mike Meredith

Examples

```
require(graphics)
oldpar <- par(mfrow=2:1)

x1 <- rnorm(1e4)            # no constraint on x1
plot(density(x1))
plot(densityFolded(x1))     # no difference

x2 <- abs(rnorm(1e4))       # x2 >= 0, with mode at 0
plot(density(x2))           # density > 0 when x2 < 0, mode around 0.2
abline(v=0, col='grey')
plot(densityFolded(x2))     # mode plotted correctly
abline(v=0, col='grey')

x3 <- rbeta(1e4, 1.5, 1.5)  # 0 <= x3 <= 1
plot(density(x3))           # density > 0 when x2 < 0 and x2 > 1
abline(v=0:1, col='grey')
plot(densityFolded(x3))
abline(v=0:1, col='grey')
```

```
x4 <- rbeta(1e4, 1.5, 0.9)  # 0 <= x4 <= 1, with mode at 1
plot(density(x4))           # mode appears to be around 0.95
abline(v=0:1, col='grey')
plot(densityFolded(x4))     # mode plotted correctly
abline(v=0:1, col='grey')

# Try with a matrix
x5 <- cbind(rbeta(1e4, 2,2), rbeta(1e4, 2,3), rbeta(1e4, 3,2))
plot(density(x5))
tmp <- densityFolded(x5)
with(tmp, matplot(x, y, type='l'))

par(oldpar)
```

`diagPlot` *Diagnostic graphics for class* `mcmcOutput`

Description

Display trace plots and density plots for the chains in the MCMC output. Each chain is plotted with a different colour.

Usage

```
diagPlot(object, params, howMany, chains,
  maxRows=4, RhatBad=1.05, precision=c("MCEpc", "n.eff"), ask=NULL, ...)

tracePlot(object, layout=c(3,3), ask=NULL, ...)
densityPlot(object, layout=c(3,3), ask=NULL, ...)
acfPlot(object, lag.max=NULL, layout=c(3,3), ask=NULL, ...)
```

Arguments

`object`	An object of any class with MCMC output that can be coerced to class `mcmcOutput`.
`params`	An optional vector of column numbers or names; names are partially matched, so `params="alpha"` will match all of `alpha`, `alpha0`, `alpha[1]`, `alphanew`; negative indices indicate parameters to exclude, positive and negative indices cannot be mixed.
`howMany`	How many draws per chain to plot; if negative, the draws at the end of the chains will be plotted; default is to plot all.
`chains`	Which chains to plot, a numeric vector; default is to plot all.
`maxRows`	Maximum number of rows to display in one window; each row consists of a trace plot and a density plot for one parameter.
`RhatBad`	Threshold for Rhat; parameters with `Rhat` > `RhatBad` are highlighted in red.
`precision`	The statistic to use for the precision, displayed above the density plot.

layout	a length-2 vector with the maximum number of rows and columns to display in the plotting frame.
lag.max	Maximum lag at which to calculate the acf.
ask	If TRUE and the number of parameters to plot is greater than maxRows, the user will be prompted before the next page of output is displayed. The default is to ask if the plotting device is the screen, not if it is a file.
...	Additional graphical parameters.

Value

Return nothing, used for their plotting side effects.

Author(s)

Mike Meredith

See Also

crosscorrPlot, postPlot for a histogram and summary statistics.

Examples

```
# Create a fake mcmcOutput object:
tmp <- cbind(
  mu0 = rnorm(3000),         # normal, mean zero
  mu10 = rnorm(3000, rep(9:11, each=1000), 1),
                             # normal, mean 10, but poor mixing
  sigma=rlnorm(3000),        # non-negative, skewed
  `prob[1,1]` = rbeta(3000, 4, 4),  # probability, central mode
  `prob[1,2]` = 0.3,         # constant
  `prob[2,1]` = rbeta(3000, 1, 3),  # probability, mode = 0
  N = rpois(3000, rep(c(24, 18, 18), each=1000)),
                             # large integers (no zeros), poor mixing
  n = rpois(3000, 2),        # small integers (some zeros)
  allNA = NA,                # all values NA
  someNA = suppressWarnings(log(rnorm(3000, 2, 2))),
                             # some NaNs
  const1 = rep(1, 3000),     # all values = 1
  const3.2 = rep(10/3, 3000))# all values the same but not integer
( fake <- mcmcOutput(tmp, nChains = 3) )
summary(fake)
diagPlot(fake)
diagPlot(fake, params=3:6, main="params = 3:6")
diagPlot(fake, params=c("mu", "prob"), main="params = c('mu', 'prob')")
diagPlot(fake, params=c("mu", "prob"), howMany=200, main="howMany = 200")
diagPlot(fake, params=c("mu", "prob"), howMany=50, main="howMany = 50")
diagPlot(fake, params=c("mu", "prob"), howMany=-200, main="howMany = -200")
diagPlot(fake, params=c("mu", "prob"), chains=1:2, main="chains = 1:2")
diagPlot(fake, params=c("mu", "prob"), chains=2, main="chains = 2") # 1 chain -> no Rhat

tracePlot(fake, layout=c(2,2))
```

```
densityPlot(fake, xlab="value")
acfPlot(fake, lag.max=10, lwd=2)

# Use diagPlot with an mcmc.list object
data(mcmcListExample)
diagPlot(mcmcListExample)
diagPlot(mcmcListExample, main="example", params=1:3, precision="n.eff")
```

`discrepancyPlot` *Graphic comparison of observed vs simulated discrepancies*

Description

One way to assess the fit of a model is to calculate the discrepancy between the observed data and the values predicted by the model. For binomial and count data, the discrepancy will not be zero because the data are integers while the predictions are continuous. To assess whether the observed discrepancy is acceptable, we simulate new data according to the model and calculate discrepancies for the simulated data.

Function `discrepancyPlot` produces a scatter plot of the MCMC chains for observed vs simulated discrepancies and calculates and displays a p-value, the proportion of simulated discrepancy values that exceed the observed discrepancy.

Usage

```
discrepancyPlot(object, observed, simulated, ...)
```

Arguments

`object`	An object of class `mcmcOutput`, or an object which can be coerced to `mcmcOutput`, with MCMC chains for observed and simulated discrepancies.
`observed`	character; the name of the parameter for the observed discrepancy.
`simulated`	character; the name of the parameter for the simulated discrepancies.
`...`	additional graphical parameters passed to the `plot.default`.

Value

Returns the proportion of simulated discrepancy values that exceed the observed discrepancy, often referred to as a "Bayesian p-value".

Author(s)

Mike Meredith.

Examples

```
# Get some data
data(mcmcListExample)
( mco <- mcmcOutput(mcmcListExample) )
# Tobs and Tsim are the Freeman-Tukey discrepancy measures

discrepancyPlot(mco, observed="Tobs", simulated="Tsim") # defaults
discrepancyPlot(mco, observed="Tobs", simulated="Tsim",
    main="Salamanders", col='red')
```

Get diagnostics *Get diagnostic statistics*

Description

These functions calculated diagnostic statistics for objects of class `mcmcOutput`. Optionally, only values that are worse than a predefined threshold will be returned, and values can be sorted so that the worst are at the start of the output vector.

Usage

```
getMCE(x, pc=TRUE, bad=5, sort=TRUE)
getNeff(x, bad=10000, sort=TRUE)
getRhat(x, bad=1.1, sort=TRUE)
```

Arguments

`x`	an object of any class with MCMC output that can be coerced to class `mcmcOutput`.
`pc`	if TRUE, the value of the MC error as a percentage of the posterior SD will be returned.
`bad`	threshold for "bad" values: only values above this (for `getMCE` or `getRhat`) or below this (for `getNeff`) will be returned. If `bad = NA`, all values will be returned, including NAs.
`sort`	if TRUE, the values will be sorted, with the worst at the top.

Details

`getRhat` returns the Brooks-Gelman-Rubin (BGR) convergence diagnostic (Brooks & Gelman 1998), a non-parametric 'interval' estimator of the 'potential scale reduction factor' for MCMC output. Similar to the function `coda::gelman.diag`, but faster when thousands of parameters are involved and will not cause R to crash.

`getMCE` returns the Monte Carlo standard error calculated using the batch method of Lunn et al (2013, p77); see also Roberts (1996).

`getNeff` returns the effective number of draws taking account of autocorrelation within each chain. It is a wrapper for `coda::effectiveSize`.

Value

A named vector with the values of the diagnostic. Values of NA will be excluded unless bad = NA. It may have length 0 if no values are bad.

Author(s)

Mike Meredith

References

Brooks, S.P. & Gelman, A. (1998) General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7, 434-455.

Lunn, D., Jackson, C., Best, N., Thomas, A., & Spiegelhalter, D. (2013) *The BUGS book: a practical introduction to Bayesian analysis*, Chapman and Hall.

Roberts, G.O. (1996). Markov chain concepts related to sampling algorithms. In *Markov Chain Monte Carlo in practice* (eds W.R. Gilks, D.J. Spiegelhalter & S. Richardson). Chapman & Hall, London.

Examples

```
data(mcmcListExample)
mco <- mcmcOutput(mcmcListExample)

getMCE(mco, bad=2)
getMCE(mco, bad=0)  # returns all except NAs
getMCE(mco, bad=NA)  # returns all including NAs
getMCE(mco, bad=NA, sort=FALSE)  # returns all, in original order

getNeff(mco, bad=2800)
getNeff(mco, bad=Inf)  # returns all except NAs
getNeff(mco, bad=NA)   # returns all including NAs

getRhat(mco)
getRhat(mco, bad=0)
getRhat(mco, bad=NA, sort=FALSE)

# Extract the values with 'bad' MCE and do plots:
( badNodes <- names(getMCE(mco, bad=2)) )
( badMco <- mco[badNodes] )
plot(badMco)
```

`mcmcListExample` *An object of class* `mcmc.list` *produced by* `rjags::coda.samples`

Description

This is the output of a basic occupancy model applied to detection/non-detection data for blue ridge salamanders (*Eurycea wilderae*) in Great Smoky Mountains National Park (MacKenzie et al, 2006 p99). Detections were recorded for 5 visits to each of 39 sites, and the data are the number of visits where the species was detected.

The model has five parameters:

psi scalar, the probability of occupancy.

p `p[1,1]` is the probability of detection given presence; `p[2,2]` is a dummy variable with values drawn from a Beta(0.5,0.5) distribution; `p[1,2]` and `p[2,1]` are not defined, so `p` is a "ragged array".

z a vector of length 39, one value for each site.

Tobs scalar, the Freeman-Tukey discrepancy for the observed data.

Tsim scalar, the Freeman-Tukey discrepancy for the simulated data.

The number of nodes monitored is 44 (`p[1,2]` and `p[2,1]` are not monitored).

Three MCMC chains were run, with 1000 adaptation iterations, 1000 burn-in, and 1000 iterations saved per chain after thinning by 10.

Usage

```
data("mcmcListExample")
```

Format

`mcmcListExample` is `mcmc.list` object as defined in package **coda**.

References

MacKenzie, D I; J D Nichols; A J Royle; K H Pollock; L L Bailey; J E Hines 2006. *Occupancy estimation and modeling : inferring patterns and dynamics of species occurrence*. Elsevier Publishing.

Examples

```
data(mcmcListExample)
str(mcmcListExample)

# convert to class mcmcOutput
( mco <- mcmcOutput(mcmcListExample) )
summary(mco)

# Extract with "$"
p <- mco$p
str(p)
p[1:5,,]  # Elements of p not defined in the model are filled with NAs

# "[" with one index, produces new mcmcOutput object
head(mco[4:5])
```

```
print(mco[c("z[35]", "z[39]")])

# "[" with two indices
mco[1:5, "psi"]  # First 5 values for psi (chain #1)

# "[" with three indices
mco[1:5, 2, "psi"] # First 5 values for psi in chain #2
```

`mcmcOutput-class` *Conversion to class* `mcmcOutput`

Description

Convert output containing MCMC chains to the class `mcmcOutput`. The function is generic, with methods for a range of input objects.

`print`, `summary`, `plot` and `window` methods are available for the class. See also `postPlot`, `discrepancyPlot`, `crosscorrPlot`, `postPriorOverlap`.

Usage

```
mcmcOutput(object, ...)

## Default S3 method:
mcmcOutput(object, ...)

## S3 method for class 'mcmc.list'
mcmcOutput(object, header, ...)

## S3 method for class 'mcmc'
mcmcOutput(object, header, ...)

## S3 method for class 'jagsUI'
mcmcOutput(object, header, ...)

## S3 method for class 'bugs'
mcmcOutput(object, header, ...)

## S3 method for class 'rjags'
mcmcOutput(object, header, ...)

## S3 method for class 'runjags'
mcmcOutput(object, header, ...)

## S3 method for class 'matrix'
mcmcOutput(object, nChains=1, header, ...)
```

```
## S3 method for class 'data.frame'
mcmcOutput(object, nChains=1, header, ...)
```

Arguments

object	an object containing the MCMC chains; see Details.
header	text to use as the header by print and summary methods.
nChains	the number of chains.
...	named arguments to be passed to other methods (currently not used).

Details

mcmcOutput objects store the output from MCMC estimation runs in a compact and easily accessible format. Several customised extraction methods are available:

$	: extracts arrays for individual parameters in the same way as a sims.list.
[with 1 index	: returns a new mcmcOutput object with the selected node(s).
[with 2 indices	: returns the selected row(s) and columns(s).
[with 3 indices	: behaves as an iterations x chains x nodes array.

Value

An object of class mcmcOutput. This is a matrix with a column for the MCMC chain for each node monitored. The first 2 attributes in the list below must be present, the rest are optional but may be used by print or summary methods:

nChains	the number of chains.
simsList	a list specifying which columns correspond to each parameter.
header	text to be displayed as the first line when the object is printed.
call	the original function call.
modelFile	the name of the original model file.
timeTaken	the time taken in seconds for the MCMC run.
runDate	an object of class POSIXct with the date of the MCMC run.

Author(s)

Mike Meredith.

---

plot.mcmcOutput *Graphic display of marginal posterior probability distributions*

Description

Plot the posterior probability distribution(s) for the nodes of a mmcmcOutput object. postPlot is equivalent to plot(mcmcOutput(object)).

Note the new argument center *with options to display the mean, median or mode; this replaces the* showMode *argument. The argument* CRImass *replaces* credMass.

Usage

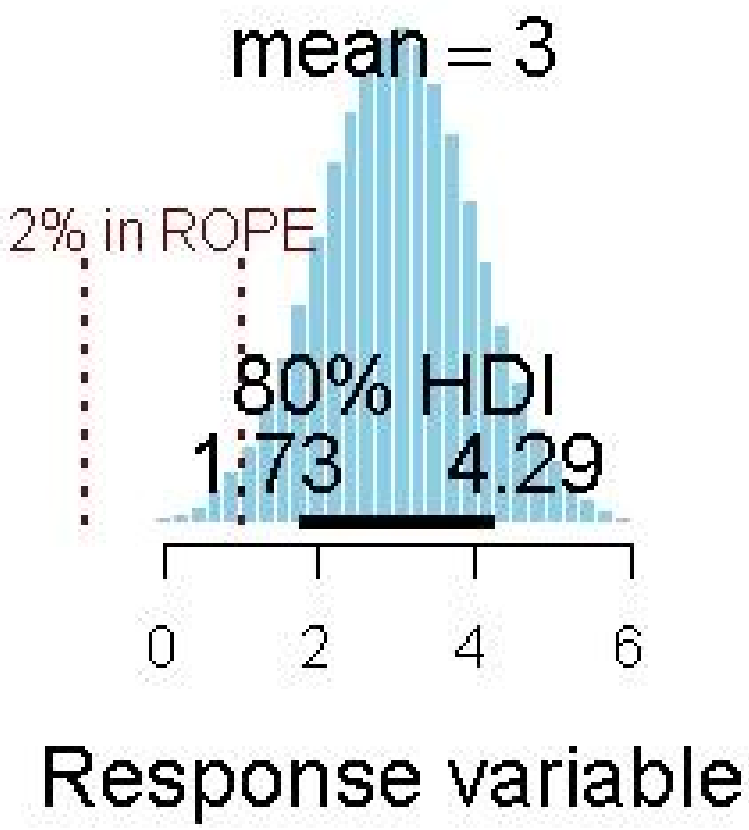
```
## S3 method for class 'mcmcOutput'
plot(x, params, layout=c(3,3),
    center = c("mean", "median", "mode"), CRImass=0.95,
    compVal = NULL, ROPE = NULL, HDItextPlace = 0.7,
    showCurve = FALSE, shadeHDI = NULL, ...)


postPlot(object, params, layout=c(3,3),
    center = c("mean", "median", "mode"), CRImass=0.95,
    compVal = NULL, ROPE = NULL, HDItextPlace = 0.7,
    showCurve = FALSE, shadeHDI = NULL, ...)
```

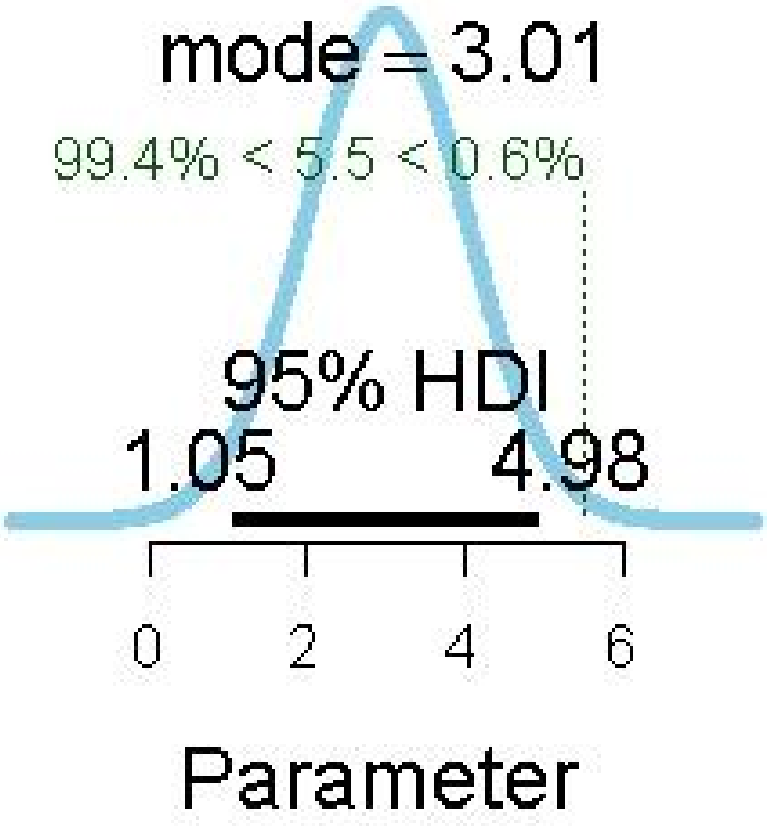
Arguments

x	An object of class mcmcOutput.
object	An object of any class that can be coerced to class mcmcOutput.
params	An optional vector of column numbers or names; names are partially matched, so params="alpha" will match all of alpha,alpha0,alpha[1],alphanew; negative indices indicate parameters to exclude, positive and negative indices cannot be mixed.
layout	a length-2 vector with the maximum number of rows and columns to display in the plotting frame.
center	the statistic to use to represent the central tendency.
CRImass	the probability mass to include in credible intervals; set this to NULL to suppress plotting of credible intervals.
compVal	a single value for comparison with those plotted; the same value will be used for all the plots.
ROPE	a two element vector, such as c(-1,1), specifying the limits of the Region Of Practical Equivalence; the same value will be used for all the plots.
HDItextPlace	a value in [0,1] that controls the horizontal position of the labels at the ends of the HDI bar.
showCurve	logical: if TRUE, the posterior density will be represented by a kernel density function instead of a histogram.
shadeHDI	specifies a colour to shade the area under the curve corresponding to the HDI; NULL for no shading. Ignored if showCurve = FALSE. Use colours() to see a list of possible colours.
...	graphical parameters and the breaks parameter for the histogram.

Details

The data are plotted either as a histogram (above) or, if showCurve = TRUE, as a fitted kernel density curve (below). The mean, median or mode of the distribution is displayed, depending on the parameter center. The Highest Density Interval (HDI) is shown as a horizontal bar, with labels for the ends of the interval.


mean = 3
2% in ROPE
80% HDI
1.73
4.29
0
2
4
6
Response variable



If a comparison value (`compVal`) is supplied, this is shown as a vertical green dotted line, together with the probability mass below and above this value. If values for a ROPE are supplied, these are shown as dark red vertical dashed lines, together with the percentage of probability mass within the ROPE.

Value

Returns nothing. Used for its plotting side-effect.

Author(s)

Mike Meredith, based on code by John Kruschke.

See Also

For details of the HDI calculation, see `hdi`.

Examples

```
# Use the example data set
data(mcmcListExample)
mco <- mcmcOutput(mcmcListExample)
plot(mco)
```

```
plot(mco, "p")   # plots p[1,1], p[2,2] and psi
plot(mco, "p[")  # plots p[1,1] and p[2,2], not psi

# Generate some data
normal <- rnorm(1e5, 2, 1)
postPlot(normal)
postPlot(normal, col='wheat', border='magenta')
postPlot(normal, CRImass=0.8, compVal=0, ROPE=c(-0.2,0.2),
    xlab="Response variable")
postPlot(normal, center="mode", showCurve=TRUE, compVal=5.5)

# For integers:
integers <- rpois(1e5, 12)
postPlot(integers)

# A severely bimodal distribution:
bimodal <- c(rnorm(1e5), rnorm(5e4, 7))
postPlot(bimodal)                  # A valid 95% CrI, but not HDI
postPlot(bimodal, showCurve=TRUE)  # Correct 95% HDI
postPlot(bimodal, showCurve=TRUE, shadeHDI='pink')
```

`postPriorOverlap` *Overlap between posterior and prior probability distributions.*

Description

Calculates and displays the overlap between a posterior distribution (as a vector of values, typically draws from an MCMC process) and a prior distribution (as a vector of values or as a function).

Usage

```
postPriorOverlap(x, prior, priorPars, breaks=NULL,
           hcols=c("skyblue", "yellow", "green", "white"), ...)
```

Arguments

`x`	a vector of values drawn from the target distribution.
`prior`	*either* a vector of values drawn from the prior distribution *or* the name for the density function of the distribution; standard R functions for this have a `d-` prefix, eg. `dbeta`.
`priorPars`	a named list of parameters to be passed to `prior` when it is a function ; see the examples. Ignored if `prior` is a numeric vector.
`breaks`	controls the histogram break points or the number of bars; see `hist`.
`hcols`	a vector of four colours for the histograms: posterior, prior, overlap, and borders. See the Color Specification section of `par`
`...`	other graphical parameters.

Value

Returns the overlap, the area lying under the lower of the two density curves.

Author(s)

Mike Meredith

Examples

```
# Generate some data
foo <- rbeta(1e4, 5, 7)

# check overlap with a Beta(0.2, 0.2) prior:
postPriorOverlap(foo, dbeta, list(shape1=0.2, shape2=0.2))

# check overlap with a Uniform(0, 1) prior:
postPriorOverlap(foo, runif(1e6))
```

---

summary.mcmcOutput *Print and summary methods for objects of class* mcmcOutput

---

Description

summary generates a data frame with summary and diagnostic statistics for each of the MCMC chains in the mcmcOutput object, and (if verbose = TRUE) prints a brief overview to the Console. The data frame will be displayed in the Console unless assigned to an object or passed to View.

sumryList generates summary and diagnostic statistics for each of the MCMC chains in the mcmcOutput object and returns them as a list-of-lists, see Value.

summary *and* sumryList *now have a* CRItype="none" *option and arguments to include* overlap0 *and* f *in the summary table.*

print displays characteristics of the mcmcOutput object. It does *not* display summaries of the values: for that use summary.

Usage

```
## S3 method for class 'mcmcOutput'
summary(object, digits=3, median=TRUE, mode=FALSE,
    CRItype=c("hdi", "symmetrical", "none"),
    CRImass=0.95, Rhat=TRUE, MCEpc = TRUE, n.eff=FALSE,
    overlap0=FALSE, f=FALSE, verbose=TRUE, ...)

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

sumryList(object, median=TRUE, mode=FALSE,
    CRItype=c("hdi", "symmetrical", "none"),
    CRImass=0.95, Rhat=TRUE, MCEpc = TRUE, n.eff=FALSE,
    overlap0=FALSE, f=FALSE, ...)
```

Arguments

x, object	an object of class mcmcOutput.
digits	the number of digits for rounding of values in the output.
median	if TRUE, the median will be included as a column in the data frame produced.
mode	if TRUE, the mode will be included as a column in the data frame produced.
CRItype	if hdi, the credible interval will be a highest density interval; if symmetrical, a symmetrical CRI will be generated with quantile; ignored if CRImass=NA.
CRImass	the probability mass to include in the credible interval; if CRImass=NA, no CRI will be included in the output.
Rhat	if TRUE, estimates of Rhat will be included; ignored if only 1 chain or < 100 values per chain. See getRhat.
MCEpc	if TRUE, estimates of the Monte Carlo standard error as a percentage of the posterior SD will be included; ignored if < 100 values per chain. See getMCE.
n.eff	if TRUE, estimates of the effective number of draws allowing for autocorrelation will be included; ignored if < 100 values per chain. See getNeff.
overlap0	if TRUE, a column with TRUE/FALSE will be included, indicating whether the credible interval includes zero. Ignored if no CRI is calculated.
f	if TRUE, a column with the proportion of the posterior with the same sign as the mean.
verbose	if FALSE, suppresses output to the Console, other than the table of statistics.
...	further arguments for the print or summary function.

Value

print returns x invisibly.

summary returns a data frame with columns for summary and diagnostic statistics and a row for each node; mean and SD are always included, other columns if selected. It has attributes for nChains and simsList derived from the input object. The output will appear in the Console unless assigned to an object or passed to View.

sumryList returns a list with a component for each statistic; each component is itself a list with the values for each parameter. See examples.

Author(s)

Mike Meredith.

Examples

```
data(mcmcListExample)
mco <- mcmcOutput(mcmcListExample)
mco  # equivalent to ...
print(mco)
summary(mco)
# just the summary data frame
summary(mco, verbose=FALSE)
```

```
# assign the output to an object
tmp <- summary(mco, median=FALSE, CRItype="sym", CRImass=0.8,
    n.eff=TRUE, MCEpc=FALSE)
tmp

mcos <- sumryList(mco)
str(mcos)
mcos$mean$p
mcos$MCEpc$z
```

`window.mcmcOutput` *Subsetting chains for* `mcmcOutput` *objects*

Description

The `window` method extracts the subset of the draws between `start` and `end`. Setting `thin = k` selects every kth observation starting from `start`.

Use `window` to discard additional draws at the start of the chain if extra burn-in is required, or to reduce the size of the object by thinning. See the examples.

Any previous burn-in or thinning is ignored (unlike the `coda::window.mcmc` method).

Usage

```
## S3 method for class 'mcmcOutput'
window(x, start=1, end=NULL, thin=1, ...)
```

Arguments

`x`	an object of class `mcmcOutput`.
`start`	the first observation to retain.
`end`	the last observation to retain; if NULL, this is set to the end of the chain.
`thin`	the interval between retained observations.
`...`	further arguments for other methods (not used).

Value

Returns an object of class `mcmcOutput` with the subset of observations.

Author(s)

Mike Meredith.

Examples

```
data(mcmcListExample)
mco <- mcmcOutput(mcmcListExample)
mco

new1 <- window(mco, start=201) # Discard the first 200 draws in each chain
new1  # Now only 800 per chain.

new2 <- window(mco, thin=3) # Retain only 1/3 of the draws
new2  # new2 is smaller; each chain reduced from 1000 to 333, total draws 999.
```

Index