Package ‘jagshelper’

November 7, 2024

Type Package

Title Extracting and Visualizing Output from 'jagsUI'

Version 0.4.1

Date 2024-11-07

Author Matt Tyers [aut, cre]

Maintainer Matt Tyers <matttyersstat@gmail.com>

Description Tools are provided to streamline Bayesian analyses in 'JAGS' using the 'jagsUI' package. Included are functions for extracting output in simpler format, functions for streamlining assessment of convergence, and functions for producing summary plots of output. Also included is a function that provides a simple template for running 'JAGS' from 'R'. Referenced materials can be found at <DOI:10.1214/ss/1177011136>.

License GPL-2

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.1

Depends R (>= 3.5.0)

Imports jagsUI, MASS

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/mbtyers/jagshelper

BugReports https://github.com/mbtyers/jagshelper/issues

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2024-11-07 18:50:01 UTC

Contents

`jagshelper-package` *Functions for Extracting and Visualizing Output from 'jagsUI'*

Description

Functions are provided to help run Bayesian analyses in JAGS using the 'jagsUI' package. Included are functions for extracting output in simpler format, functions for streamlining assessment of convergence, and functions for producing summary plots of output. Also included is a function that provides a simple template for running JAGS from R.

Details

Package:	jagshelper
Type:	Package
Version:	0.4.1
Date:	2024-11-07
License:	GPL-2

The jagshelper package is intended to extend and streamline Bayesian analysis using the 'jagsUI' package.

The skeleton function prints a template JAGS model with associated R code to the console, which can easily be copied & pasted to an R script and modified as needed.

Functions are also provided for visually assessing model convergence. In particular, tracedens_jags gives a relatively simple syntax for trace plots of a collection or subset of parameter nodes, and overlays by-chain kernel densities for visual assessment of marginal posterior shapes as well as overlap between MCMC chains. Another function that could be particularly useful to users is plotRhats, which gives a visual representation of the values of the Gelman-Rubin convergence diagnostic `Rhat` (or alternately effective sample size `n.eff`) for all saved parameters. This may be particularly useful in the case where a model has many saved parameters. Additionally, function traceworstRhat is a wrapper for tracedens_jags, but only produces trace plots for the parameter nodes with the worst (largest) values of `Rhat` or `n.eff`. Functions qq_postpred, ts_postpred, and plot_postpred provide some posterior predictive checks of a vector of data and corresponding vector (matrix, in output form) of posterior predictive samples. Function kfold provides automated k-fold or leave-one-out cross validation, giving a quick means of comparison of predictive power between candidate models.

Functions are also provided for visualizing posterior densities; in particular, the case of a vector of parameter nodes (one-dimensional in the JAGS model, giving a two-dimensional matrix of MCMC iterations). Notably, the envelope function is intended for a sequence of nodes (as in a time series), and the caterpillar function is intended for cases in which order may not matter (as in a collection of random effects). The crossplot function provides methods for bivariate plotting of two parameters, or for overlaying paired nodes of two parameter vectors.

Wrapper functions are also given for overlay of multiple such plots, as overlayenvelope and comparecat, and comparedens giving plots as vertically-oriented left- and right-facing kernel densities.

Author(s)

Matt Tyers

Maintainer: Matt Tyers <matttyersstat@gmail.com>

`asdf_jags_out` *Example data: asdf jags out*

Description

A simple model, equivalent to that produced by the output produced by `\link{skeleton}`.

Usage

```
asdf_jags_out
```

Format

An object of class `jagsUI` of length 24.

`asdf_prior_jags_out` *Example data: asdf prior jags out*

Description

A simple model, equivalent to that produced by the output produced by `\link{skeleton}`, with the addition of prior samples for all parameters.

Usage

```
asdf_prior_jags_out
```

Format

An object of class `jagsUI` of length 24.

caterpillar *Caterpillar plot*

Description

Caterpillar plot of the posterior densities of a vector of parameter nodes, in which the sequential order of nodes might not be important, such as vector of random effects.

This produces a set of overlayed interval bars (default values are 50 percent and 95 percent), with overlayed median markings, for each of a vector of parameter nodes.

Usage

```
caterpillar(
  df,
  p = NULL,
  x = NA,
  row = NULL,
  column = NULL,
  median = TRUE,
  mean = FALSE,
  ci = c(0.5, 0.95),
  lwd = 1,
  col = 4,
  add = FALSE,
  xlab = "",
  ylab = "",
  main = NULL,
  ylim = NULL,
  xax = NA,
  transform = c("none", "exp", "expit"),
  medlwd = lwd,
  medwd = 1,
  ...
)
```

Arguments

`df`	Output object returned from `jagsUI::jags()`; or alternately, two-dimensional `data.frame` or matrix in which parameter node element is given by column and MCMC iteration is given by row. A vector may also be used, that expresses MCMC iterations of a single parameter node.
`p`	Parameter name, if input to `df` is a `jagsUI` output object.
`x`	Vector of X-coordinates for plotting.
`row`	Row to subset, in the case of a 2-d matrix of parameter nodes in-model.
`column`	Column to subset, in the case of a 2-d matrix of parameter nodes in-model.

`median`	Whether to include medians
`mean`	Whether to include means
`ci`	Vector of intervals to overlay. Defaults to 50 percent and 95 percent.
`lwd`	Base line width for plotting. Defaults to 1.
`col`	Color for plotting
`add`	Whether to add to existing plot
`xlab`	X-axis label
`ylab`	Y-axis label
`main`	Plot title. If the default (`NULL`) is accepted and argument `p` is used, `p` will be used for the title.
`ylim`	Y-axis limits. If the default (`NULL`) is accepted, the limits will be determined automatically.
`xax`	Vector of possible x-axis tick labels. Defaults to the `data.frame` column names.
`transform`	Should the y-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transform="exp"`is used, consider adding additional plotting argument `log="y"`.
`medlwd`	Line width of median line
`medwd`	Relative width of median line. Defaults to 1, perhaps smaller numbers will look better?
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

envelope, crossplot

Examples

```
## usage with input data.frame
a <- jags_df(asdf_jags_out, p="a")

caterpillar(a)
caterpillar(a, ci=seq(.1,.9,by=.1))
caterpillar(a, lwd=2)
caterpillar(a, xax=c("effect 1", "effect 2", "effect 3"))


## usage with input as jagsUI object
```

```
caterpillar(asdf_jags_out, p="a")
caterpillar(SS_out, p="rate")

## usage with a 2-d parameter matrix
caterpillar(SS_out, p="cycle_s", column=1)
caterpillar(SS_out, p="cycle_s", column=2)

## usage with an exponential transformation
caterpillar(SS_out, p="trend", transform="exp", ylab="exp transform")
caterpillar(SS_out, p="trend", transform="exp", ylab="exp transform", log="y")
caterpillar(SS_out, p="trend", transform="expit", ylab="expit (inv logit) transform")
```

`chaindens_df` *By-chain kernel density of each column of a* `data.frame`*.*

Description

By-chain kernel density plot of each column of a posterior `data.frame`.

Usage

```
chaindens_df(df, nline, parmfrow = NULL, ...)
```

Arguments

`df`	Posterior `data.frame`
`nline`	Number of chains
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`...`	additional plotting arguments or arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

tracedens_jags, trace_jags, trace_line

Examples

```
a <- jags_df(asdf_jags_out, p="a")

chaindens_df(a, nline=3, parmfrow=c(3,1))
```

chaindens_jags — *By-chain kernel densities of* `jagsUI` *object*

Description

By-chain kernel densities of a whole `jagsUI` object, or optional subset of parameter nodes.

Usage

```
chaindens_jags(x, p = NULL, exact = FALSE, parmfrow = NULL, lwd = 1, ...)
```

Arguments

`x`	Posterior `jagsUI` object
`p`	Parameter name for subsetting: if this is specified, only parameters with names beginning with this string will be plotted.
`exact`	Whether `p` should be an exact match (`TRUE`) or just match the beginning of the string (`FALSE`). Defaults to `FALSE`.
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`lwd`	Line width for plotting. Defaults to 1.
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

tracedens_jags, trace_jags, chaindens_line, chaindens_df

Examples

```
chaindens_jags(asdf_jags_out, parmfrow=c(4,2))
chaindens_jags(x=asdf_jags_out, p="a", parmfrow=c(3,1))
```

`chaindens_line` *Simple by-chain kernel density plot*

Description

By-chain kernel density plot of a single parameter node.

Usage

```
chaindens_line(x, nline, lwd = 1, main = "", ...)
```

Arguments

`x`	Posterior vector
`nline`	Number of chains
`lwd`	Line width
`main`	Plot title
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

tracedens_jags, chaindens_jags, chaindens_df

Examples

```
b1 <- jags_df(asdf_jags_out, p="b1")

chaindens_line(b1, nline=3, main="b1")
```

`check_neff` *Quick summary of n.eff values by parameter name*

Description

Returns the mean number of `n.eff` values (by each parameter) that are greater than a specified threshold criterion.

`n.eff` is calculated within 'JAGS', and may be interpreted as a crude measure of effective sample size for a given parameter node.

Usage

```
check_neff(x, thresh = 500)
```

Arguments

`x`	Output object from `jagsUI::jags()`
`thresh`	Threshold value (defaults to 500)

Value

Numeric (named) giving the proportion of `n.eff` values above the given threshold.

Author(s)

Matt Tyers

See Also

check_Rhat, traceworstRhat, plotRhats, qq_postpred, ts_postpred

Examples

```
check_neff(SS_out)
```

`check_Rhat` *Quick summary of Rhat values by parameter name*

Description

Returns the mean number of `Rhat` values for each parameter (by each parameter) that are less than a specified threshold criterion.

`Rhat` (Gelman-Rubin Convergence Diagnostic, or Potential Scale Reduction Factor) is calculated within 'JAGS', and is commonly used as a measure of convergence for a given parameter node. Values close to 1 are seen as evidence of adequate convergence.

Usage

```
check_Rhat(x, thresh = 1.1)
```

Arguments

`x`	Output object from `jagsUI::jags()`
`thresh`	Threshold value (defaults to 1.1)

Value

Numeric (named) giving the proportion of Rhat values below the given threshold.

Author(s)

Matt Tyers

References

Gelman, A., & Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science, 7*(4), 457–472. http://www.jstor.org/stable/2246093

See Also

check_neff, traceworstRhat, plotRhats, qq_postpred, ts_postpred

Examples

```
check_Rhat(SS_out)
```

`comparecat` *Compare Caterpillar Plots*

Description

Interleaved caterpillar plots for all parameters (or a specified subset) from a list of `jagsUI` output objects or `data.frames`. The intent of this function is easy comparison of inferences from multiple comparable models.

Here a caterpillar plot is defined as a set of overlayed interval bars (default values are 50 percent and 95 percent), with overlayed median markings, for each of a vector of parameter nodes.

Usage

```
comparecat(
  x,
  p = NULL,
  ci = c(0.5, 0.95),
  ylim = NULL,
  col = NULL,
  xlab = "",
  ylab = "",
  transform = c("none", "exp", "expit"),
  ...
)
```

Arguments

`x`	List of output objects returned from `jagsUI` or `data.frames`
`p`	Optional vector of parameters to subset. All parameters with names matching the beginning of the string supplied will be returned. If the default (`NULL`) is accepted, all parameters will be plotted.
`ci`	Credible intervals widths to plot. Defaults to 50% and 95%.
`ylim`	Y-axis limits for plotting. If the default (`NULL`) is accepted, limits will be automatically determined.
`col`	Vector of colors for plotting. If the default (`NULL`) is accepted, colors will be automatically drawn.
`xlab`	X-axis label
`ylab`	Y-axis label
`transform`	Should the y-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transform="exp"`is used, consider adding additional plotting argument `log="y"`.
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

caterpillar, crossplot, comparedens, comparepriors

Examples

```
## This is the same output object three times, but shows functionality.
comparecat(x=list(asdf_jags_out, asdf_jags_out, asdf_jags_out),
           p=c("a","b","sig"))

## Transformed
comparecat(x=list(asdf_jags_out, asdf_jags_out, asdf_jags_out),
           p=c("sig"), transform="exp")
comparecat(x=list(asdf_jags_out, asdf_jags_out, asdf_jags_out),
           p=c("sig"), transform="exp", log="y")
```

---

comparedens *Compare Density*

Description

Side-by-side kernel density plots for all parameters (or a specified subset) from two `jagsUI` output objects or `data.frames`. The intent of this function is easy comparison of inferences from two comparable models.

Kernel densities are plotted vertically, either left- or right-facing. Parameters with the same name are plotted facing one another.

Usage

```
comparedens(
  x1,
  x2,
  p = NULL,
  minCI = 0.99,
  ylim = NULL,
  legendnames = NULL,
  legendpos = "topleft",
  col = c(4, 2),
  ...
)
```

Arguments

x1	Output object returned from `jagsUI`; or alternately a `data.frame`
x2	Second output object returned from `jagsUI`; or alternately a `data.frame`
p	Optional vector of parameters to subset. All parameters with names matching the beginning of the string supplied will be returned. If the default (`NULL`) is accepted, all parameters will be plotted.
minCI	Minimum CI width for plotting. This is intended as a method for excluding far-outlying MCMC samples when determining the appropriate y-axis limits for plotting. Defaults to 99%.

`ylim`	Y-axis limits for plotting. If the default (`NULL`) is accepted, limits will be automatically determined.
`legendnames`	Names for optional legend. If the default `NULL` is accepted, no legend will be drawn.
`legendpos`	Position for optional legend. Defaults to `"topleft"`.
`col`	Colors for kernel density plots. Defaults to colors `4` and `2` (blue and red).
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

comparecat, comparepriors

Examples

```
## This is the same output object twice, but shows functionality.
comparedens(x1=asdf_jags_out, x2=asdf_jags_out, p=c("a","b","sig"),
            legendnames=c("Model 1", "Model 2"))
```

`comparepriors` *Compare Priors*

Description

Side-by-side kernel density plots for all parameters with parameter names ending in `"_prior"`, and corresponding parameters without. It should be noted that these parameters must be specified in JAGS as well as the corresponding parameters, and this is left to the user.

This function is a wrapper of comparedens.

Kernel densities are plotted vertically, either left- or right-facing. Parameters with the same name are plotted facing one another.

Usage

```
comparepriors(x, parmfrow = NULL, ...)
```

Arguments

`x`	Output object returned from jagsUI::jags()
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`...`	additional arguments to comparedens

Value

`NULL`

Author(s)

Matt Tyers

See Also

comparecat, comparedens, plotdens

Examples

```
## a look at what parameters exist in the input object
nbyname(asdf_prior_jags_out)

## then, showing the function usage
comparepriors(asdf_prior_jags_out, parmfrow=c(2, 3))
```

`cor_jags` *Correlation matrix from a JAGS object*

Description

Computes a correlation matrix of all MCMC samples from an object returned by 'jagsUI', or an optional subset of parameter nodes.

Usage

```
cor_jags(x, p = NULL, exact = FALSE)
```

Arguments

`x`	Output object returned from `jagsUI`
`p`	Optional string to begin posterior names. If `NULL` is used, all parameters will be used
`exact`	Whether name must be an exact match (`TRUE`) or with initial sub-string matching only supplied characters (`FALSE`). Defaults to `FALSE`.

Value

A 2-dimensional correlation matrix (n X n, where n is the number of parameter nodes)

Author(s)

Matt Tyers

See Also

plotcor_jags

Examples

```
cor_jags(asdf_jags_out)
```

`crossplot` *Bivariate Plot of Posterior Densities*

Description

Bivariate plot of the posterior densities of corresponding vectors of parameter nodes. Three plotting methods are provided, that may be overlayed if desired.

- If `drawcross == TRUE`, caterpillar-like plots will be produced, with quantile intervals in the x- and y- directions.
- If `drawx == TRUE`, caterpillar-like plots will be produced, but rotated along the standardized principal component axes. This may be useful to draw if correlation is present.
- If `drawblob == TRUE`, smoothed polygons will be produced, each containing approximately `ci=` x100% of the associated MCMC samples.

All methods can overlay multiple bars or polygons, depending on the length of `ci=`.

Usage

```
crossplot(
  dfx,
  dfy = NULL,
  p = NULL,
  col = 4,
  drawcross = TRUE,
  drawx = FALSE,
  drawblob = FALSE,
  blobres = NULL,
  blobsmooth = NULL,
  outline = FALSE,
  ci = c(0.5, 0.95),
  lwd = 1,
  mean = FALSE,
  link = FALSE,
  linklwd = 1,
  labels = FALSE,
  labelpos = NULL,
  labelcex = 0.7,
  whichx = NULL,
  rowx = NULL,
  columnx = NULL,
  whichy = NULL,
  rowy = NULL,
  columny = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  xlim = NULL,
  ylim = NULL,
  transformx = c("none", "exp", "expit"),
  transformy = c("none", "exp", "expit"),
  add = FALSE,
  ...
)
```

Arguments

`dfx`	Output object returned from `jagsUI::jags()`; or alternately, two-dimensional `data.frame` or matrix in which parameter node element is given by column and MCMC iteration is given by row. A vector may also be used, that expresses MCMC iterations of a single parameter node. If used with `dfy=`, this will be plotted in the x-direction.
`dfy`	Optionally, a two-dimensional `data.frame` or matrix in which parameter node element is given by column and MCMC iteration is given by row. A vector may also be used, that expresses MCMC iterations of a single parameter node. If used, this will be plotted in the y-direction.
`p`	Vector of parameter names, if input to `dfx` is a `jagsUI` output object. If used, this must be of length 2.
`col`	Color for plotting, or recyclable vector of colors. Defaults to `4`. If `col == "random"`, rcolors will be used to generate a random vector of colors.
`drawcross`	Whether to draw quantile bars in the x- and y-directions. Defaults to `TRUE`.
`drawx`	Whether to draw quantile bars along the standardized principal component axes. Defaults to `FALSE`.
`drawblob`	Whether to draw smoothed quantile polygons. Defaults to `FALSE`.
`blobres`	Optional tuning parameter for drawing quantile polygons, and corresponds to the number of polygon vertices. If the default `NULL` is accepted, the function will supply a value based on the number of MCMC samples.
`blobsmooth`	Optional tuning parameter for drawing quantile polygons, and corresponds to half the number of polygon vertices used for local smoothing. If the default `NULL` is accepted, the function will supply a value based on the number of MCMC samples and the number of vertices.
`outline`	Whether to draw quantile polygons as lines rather than filled regions. Defaults to `FALSE`.
`ci`	Vector of intervals to overlay. Defaults to 50 percent and 95 percent.
`lwd`	Base line width for plotting. Defaults to 1.

`mean`	Whether to include points for means. Defaults to `FALSE`.
`link`	Whether to link medians in sequence. Defaults to `FALSE`.
`linklwd`	Line width to use for linking. Defaults to `1`.
`labels`	Whether to add labels, or a vector of labels to add. Defaults to `FALSE`.
`labelpos`	Optionally, an argument to `pos` in text for labels. Defaults to `NULL`.
`labelcex`	Optional character expansion for labels. Defaults to `0.7`.
`whichx`	Element to subset for x, if only one element of a vector of parameter nodes is desired for plotting.
`rowx`	Row to subset for x, in the case of a 2-d matrix of parameter nodes in-model.
`columnx`	Column to subset for x, in the case of a 2-d matrix of parameter nodes in-model.
`whichy`	Element to subset for x, if only one element of a vector of parameter nodes is desired for plotting.
`rowy`	Row to subset for y, in the case of a 2-d matrix of parameter nodes in-model.
`columny`	Column to subset for y, in the case of a 2-d matrix of parameter nodes in-model.
`xlab`	X-axis label. If the default `NULL` is accepted, this will be drawn automatically.
`ylab`	Y-axis label. If the default `NULL` is accepted, this will be drawn automatically.
`main`	Plot title.
`xlim`	X-axis limits. If the default (`NULL`) is accepted, the limits will be determined automatically.
`ylim`	Y-axis limits. If the default (`NULL`) is accepted, the limits will be determined automatically.
`transformx`	Should the x-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transformx="exp"`is used, consider adding additional plotting argument `log="x"` or `log="xy"`.
`transformy`	Should the y-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transformy="exp"`is used, consider adding additional plotting argument `log="y"` or `log="xy"`.
`add`	Whether to add to existing plot
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

caterpillar, pairstrace_jags

Examples

```
## basic functionality with cross geometry
crossplot(SS_out, p=c("trend","rate"))

## default labels
crossplot(SS_out, p=c("trend","cycle"), labels=TRUE)

## showing:
## - link lines
## - blob geometry (smoothed confidence polygons)
## - random colors with col="random"
crossplot(SS_out, p=c("trend","cycle"),
          labels=SS_data$x, labelpos=1, link=TRUE, drawblob=TRUE,
          col="random")

## adding x geometry and showing usage with a single vector element (41)
crossplot(SS_out, p=c("trend","cycle"),
          whichx=41, whichy=41,
          drawblob=TRUE, drawx=TRUE)

## single vectors (or data.frames or 2d matrices) can also be used
xx <- SS_out$sims.list$trend[,41]
yy <- SS_out$sims.list$cycle[,41]

par(mfrow = c(2, 2))
plot(xx, yy, col=adjustcolor(1, alpha.f=.1), pch=16, main="cross geometry")
crossplot(xx, yy, add=TRUE, col=1)
plot(xx, yy, col=adjustcolor(1, alpha.f=.1), pch=16, main="x geometry")
crossplot(xx, yy, add=TRUE, col=1,
          drawcross=FALSE, drawx=TRUE)
plot(xx, yy, col=adjustcolor(1, alpha.f=.1), pch=16, main="blob geometry")
crossplot(xx, yy, add=TRUE, col=1,
          drawcross=FALSE, drawblob=TRUE)
plot(xx, yy, col=adjustcolor(1, alpha.f=.1), pch=16, main="blob outlines")
crossplot(xx, yy, add=TRUE, col=1,
          drawcross=FALSE, drawblob=TRUE, outline=TRUE)
```

`envelope` *Envelope plot*

Description

Envelope plot of the posterior densities of a vector of parameter nodes, in which the sequential order of nodes is important, such as a time series.

This produces a plot of overlayed shaded strips, each corresponding to a given interval width (defaults to 50 percent and 95 percent), with an overlayed median line.

Usage

```
envelope(
  df,
  p = NULL,
  x = NA,
  row = NULL,
  column = NULL,
  median = TRUE,
  ci = c(0.5, 0.95),
  col = 4,
  add = FALSE,
  dark = 0.3,
  outline = FALSE,
  xlab = "",
  ylab = "",
  main = NULL,
  ylim = NULL,
  transform = c("none", "exp", "expit"),
  ...
)
```

Arguments

`df`	Output object returned from `jagsUI::jags()`; or alternately, two-dimensional `data.frame` or matrix in which parameter node element is given by column and MCMC iteration is given by row.
`p`	Parameter name, if input to `df` is a `jagsUI` output object.
`x`	Vector of X-coordinates for plotting.
`row`	Row to subset, in the case of a 2-d matrix of parameter nodes in-model.
`column`	Column to subset, in the case of a 2-d matrix of parameter nodes in-model.
`median`	Whether to include median line
`ci`	Vector of intervals to overlay. Defaults to 50 percent and 95 percent.
`col`	Color for plotting
`add`	Whether to add to existing plot
`dark`	Opacity (0-1) for envelopes. Note that multiple overlapping intervals will darken the envelope.
`outline`	Whether to just envelope outlines
`xlab`	X-axis label
`ylab`	Y-axis label
`main`	Plot title. If the default (`NULL`) is accepted and argument p is used, p will be used for the title.
`ylim`	Y-axis limits for plotting. If the default (`NULL`) is accepted, these will be determined automatically.

`transform`	Should the y-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transform="exp"`is used, consider adding additional plotting argument `log="y"`.
`...`	additional plotting arguments or arguments to `lines()`

Value

`NULL`

Author(s)

Matt Tyers

See Also

overlayenvelope, caterpillar

Examples

```
## usage with input data.frame
trend <- jags_df(SS_out, p="trend")
envelope(trend, x=SS_data$x)

## usage with jagsUI object
envelope(SS_out, p="trend")

## usage with 2-d jagsUI object
envelope(SS_out, p="cycle_s", column=1, main="cycle")
envelope(SS_out, p="cycle_s", column=2, col=2, add=TRUE)  ## overlay

## scale transformation
envelope(SS_out, p="trend", transform="exp", ylab="exp transform")
envelope(SS_out, p="trend", transform="exp", ylab="exp transform", log="y")
```

`expit` *Expit, or inverse logit*

Description

Inverse logit, where logit is defined as log(x/(1-x)).

Expit (inverse logit) is defined as exp(x)/(1+exp(x)).

Usage

```
expit(x)
```

Arguments

`x`	Numeric vector

Value

Numeric vector

Author(s)

Matt Tyers

See Also

logit

Examples

```
expit(0)
```

`jags_df` *Extract data.frame*

Description

Extracts the posterior samples from `jagsUI` output in the form of a `data.frame`. This simpler construction has a few benefits: operations may be more straightforward, and posterior objects will be smaller files and can be written to an external table or .csv, etc.

Usage

```
jags_df(x, p = NULL, exact = FALSE)
```

Arguments

`x`	Output object from `jagsUI::jags()`
`p`	Optional string to begin posterior names. If `NULL` is used, all parameters will be returned.
`exact`	Whether name must be an exact match (`TRUE`) or with initial sub-string matching only supplied characters (`FALSE`). Defaults to `FALSE`.

Value

A `data.frame` with a column associated with each parameter and a row associated with each MCMC iteration.

Author(s)

Matt Tyers

See Also

pull_post

Examples

```
out_df <- jags_df(asdf_jags_out)
```

`jags_plist` *Plist*

Description

Extracts a list of matrices, one for each saved parameter node. Each list element will be all posterior samples from that parameter node, arranged in a matrix with a column associated with each MCMC chain and a row for each MCMC iteration.

Usage

```
jags_plist(x, p = NULL, exact = FALSE)
```

Arguments

`x`	`jagsUI` output object
`p`	String to subset parameter names, if a subset is desired
`exact`	Whether `p` should be an exact match (`TRUE`) or just match the beginning of the string (`FALSE`). Defaults to `FALSE`.

Value

A `list` with an element associated with each parameter. Each element will be a matrix with a column associated with each MCMC chain and a row for each MCMC iteration.

Note

It is unlikely that a user will need this function; it is included primarily as a helper function used by other functions in this package.

Author(s)

Matt Tyers

Examples

```
out_plist <- jags_plist(asdf_jags_out)
str(out_plist)

a_plist <- jags_plist(asdf_jags_out, p=c("a","sig_a"))
str(a_plist)
```

`kfold` *Automated K-fold or Leave One Out Cross Validation*

Description

Runs k-fold or Leave One Out Cross Validation for a specified component of a JAGS data object, for a specified JAGS model.

JAGS is run internally `k` times (or alternately, the size of the dataset), withholding each of `k` "folds" of the input data and drawing posterior predictive samples corresponding to the withheld data, which can then be compared to the input data to assess model predictive power.

Global measures of predictive power are provided in output: Root Mean Square (Prediction) Error and Mean Absolute (Prediction) Error. However, it is likely that these measures will not be meaningful by themselves; rather, as a metric for scoring a set of candidate models.

Usage

```
kfold(
  model.file,
  data,
  p,
  addl_p = NULL,
  save_postpred = FALSE,
  k = 10,
  loocv = FALSE,
  fold_dims = NULL,
  ...
)
```

Arguments

`model.file`	Path to file containing the model written in BUGS code, passed directly to jags.
`data`	The named list of data objects, passed directly to jags.
`p`	The name of the data object to use for K-fold or LOO CV.
`addl_p`	Names of additional parameters to save from JAGS output, if a metric such as Log Pointwise Predictive Density is to be calculated from cross-validation results. Defaults to `NULL`, indicating no additional parameters.
`save_postpred`	Whether to save all posterior predictive samples, in addition to posterior medians. Defaults to `FALSE`.
`k`	How many folds to use for cross-validation. Defaults to `10`. If this is set to a number equal to (or greater than) the sample size, LOOCV behavior will result.
`loocv`	Whether to perform Leave One Out (rather than k-fold) Cross Validation. Setting this to `TRUE` will override the input to `k=`. Defaults to `FALSE`.

`fold_dims`	A vector of margins to use for selecting folds, if the data object used for cross validation is a matrix or array. For example, if the data consists of a two-dimensional matrix, setting `fold_dims=1` will result in whole rows being selected in each fold, or setting `fold_dims=2` will result in whole columns. However, this is generalized to accept vectors of multiple `fold_dims` and higher-dimensional arrays of data.
`...`	additional arguments to jags. These may (or must) include `n.chains`, `n.iter`, `n.burnin`, `n.thin`, `parallel`, etc.

Value

A named list, which may consist of the following:

- `$pred_y`: Point estimates of predicted values corresponding to each data element, calculated as the posterior predictive median value
- `$data_y`: Original data used for cross validation
- `$postpred_y`: All posterior predictive samples corresponding to each data element, if `save_postpred=TRUE`
- `$rmse_pred`: Root Mean Square (Prediction) Error
- `$mae_pred`: Mean Absolute (Prediction) Error
- `$addl_p`: A list with length equal to `k` (or the number of folds), with each list element containing all posterior samples for additional parameters, if these are supplied in argument `addl_p=`.
- `$fold`: A vector, matrix, or array corresponding to the original data, giving the numerical values of the corresponding fold used

Author(s)

Matt Tyers

See Also

qq_postpred, plot_postpred, plotRhats, traceworstRhat

Examples

```
#### test case where y is a matrix
asdf_jags <- tempfile()
cat('model {
  for(i in 1:n) {
    for(j in 1:ngrp) {
      y[i,j] ~ dnorm(mu[i,j], tau)
      mu[i,j] <- b0 + b1*x[i,j] + a[j]
    }
  }

  for(j in 1:ngrp) {
    a[j] ~ dnorm(0, tau_a)
  }

  tau <- pow(sig, -2)
  sig ~ dunif(0, 10)
```

```
    b0 ~ dnorm(0, 0.001)
    b1 ~ dnorm(0, 0.001)

    tau_a <- pow(sig_a, -2)
    sig_a ~ dunif(0, 10)
  }', file=asdf_jags)


  # simulate data to go with the example model
  n <- 45
  x <- matrix(rnorm(n, sd=3),
              nrow=20, ncol=3)
  y <- matrix(rnorm(n, mean=rep(1:3, each=20)-x),
              nrow=20, ncol=3)

  asdf_data <- list(x=x,
                    y=y,
                    n=nrow(x),
                    ngrp=ncol(x))

  # JAGS controls
  niter <- 1000
  ncores <- 2
  # ncores <- min(10, parallel::detectCores()-1)

  ## random assignment of folds
  kfold1 <- kfold(p="y",
                  k=5,
                  model.file=asdf_jags, data=asdf_data,
                  n.chains=ncores, n.iter=niter,
                  n.burnin=niter/2, n.thin=niter/1000,
                  parallel=FALSE)
  str(kfold1)
  kfold1$fold

  ## Performing LOOCV, but assigning folds by row of input data
  kfold2 <- kfold(p="y",
                  loocv=TRUE, fold_dims=1,
                  model.file=asdf_jags, data=asdf_data,
                  n.chains=ncores, n.iter=niter,
                  n.burnin=niter/2, n.thin=niter/1000,
                  parallel=FALSE)
  str(kfold2)
  kfold2$fold
```

logit *Logit*

Description

Logit log(x/(1-x))

Usage

```
logit(x)
```

Arguments

`x`	Numeric vector

Value

Numeric vector

Author(s)

Matt Tyers

See Also

expit

Examples

```
logit(0.5)
```

`nbyname` *Number of parameter nodes by parameter name*

Description

Returns a list of the numbers of parameter nodes saved in `jagsUI` output, by parameter name. As a default, what is returned for each list element is a vector of the array dimensions within the JAGS model (that is, excluding the dimension associated with the number of MCMC samples for each parameter node), or alternately, just the total number of parameter nodes.

Usage

```
nbyname(x, justtotal = FALSE)
```

Arguments

`x`	Output object from `jagsUI::jags()`
`justtotal`	Whether to just report the total number of parameters, as opposed to dimensions.

Value

A `list` with an element associated with each parameter. Each element can be interpreted as the vector length or array dimension associated with the given parameter.

Author(s)

Matt Tyers

See Also

nparam

Examples

```
head(jags_df(asdf_jags_out))

nbyname(asdf_jags_out)

nparam(SS_out)
nbyname(SS_out)
```

---

nparam *Number of parameters*

Description

Total number of individual parameter nodes saved in `jagsUI` output.

Usage

```
nparam(x)
```

Arguments

`x`	Output object from `jagsUI::jags()`

Value

A single numeric value giving the number of parameter nodes.

Author(s)

Matt Tyers

See Also

nbyname

Examples

```
head(jags_df(asdf_jags_out))

nparam(asdf_jags_out)
```

overlayenvelope *Overlay envelope plots*

Description

Overlays multiple envelope plots of posterior `data.frames`, or outputs returned from `jagsUI`. This would be best suited to a set of posterior `data.frames` or 2-d matrices representing sequential vectors of parameter nodes.

Here a single envelope plot is defined as a set of overlayed shaded strips, each corresponding to a given interval width (defaults to 50 percent and 95 percent), with an overlayed median line.

Usage

```
overlayenvelope(
  df,
  p = NULL,
  x = NA,
  row = NULL,
  column = NULL,
  median = TRUE,
  ci = c(0.5, 0.95),
  col = NULL,
  add = FALSE,
  dark = 0.3,
  outline = FALSE,
  xlab = "",
  ylab = "",
  main = NULL,
  ylim = NULL,
  legend = TRUE,
  legendnames = NULL,
  legendpos = "topleft",
  transform = c("none", "exp", "expit"),
  ...
)
```

Arguments

`df`	Primary input can be specified in a number of ways: either a `list()` of posterior `data.frames` or matrices, a `list` of output objects returned from `jagsUI::jags()`, a 3-dimensional `array` in which the input matrices to plot are separated according to the 3rd array dimension, or a single output object returned from `jagsUI::jags()` with multiple arguments passed to `p`, following.
`p`	Parameter name, if input to `df` is a list of `jagsUI` output objects; or a vector of parameter names, if input to `df` is a single `jagsUI` output object.
`x`	Optional vector of X-coordinates for plotting.

`row`	Row to subset, in the case of a 2-d matrix of parameter nodes in-model.
`column`	Column to subset, in the case of a 2-d matrix of parameter nodes in-model.
`median`	Whether to include median line
`ci`	Vector of intervals to overlay. Defaults to 50 percent and 95 percent.
`col`	Vector of colors for plotting
`add`	Whether to add to existing plot
`dark`	Opacity (0-1) for envelopes. Note that multiple overlapping intervals will darken the envelope. Defaults to 0.3.
`outline`	Whether to just envelope outlines
`xlab`	X-axis label
`ylab`	Y-axis label
`main`	Plot title. If the default (`NULL`) is accepted and argument `p=` is used, `p` will be used for the title.
`ylim`	Y-axis limits for plotting. If the default (`NULL`) is accepted, these will be determined automatically.
`legend`	Whether to automatically try to add a legend. Defaults to `TRUE`.
`legendnames`	Optional vector of names for a legend.
`legendpos`	Position for optional legend. Defaults to `"topleft"`.
`transform`	Should the y-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transform="exp"`is used, consider adding additional plotting argument `log="y"`.
`...`	additional plotting arguments or arguments to `lines()`

Value

`NULL`

Author(s)

Matt Tyers

See Also

envelope, crossplot

Examples

```
## usage with list of input data.frames
overlayenvelope(df=list(SS_out$sims.list$cycle_s[,,1],
                             SS_out$sims.list$cycle_s[,,2]))

## usage with a 3-d input array
overlayenvelope(df=SS_out$sims.list$cycle_s)
```

```
## usage with a jagsUI output object and parameter name (2-d parameter)
overlayenvelope(df=SS_out, p="cycle_s")

## usage with a single jagsUI output object and multiple parameters
overlayenvelope(df=SS_out, p=c("trend","rate"))

## exponential transformation
overlayenvelope(df=SS_out, p="cycle_s", transform="exp",
                ylab="exp transform")
overlayenvelope(df=SS_out, p="cycle_s", transform="exp",
                ylab="exp transform", log="y")
```

`pairstrace_jags` *Pairs trace plot*

Description

Two-dimensional trace plots (or alternately, scatter plots or contour plots) of each possible pair of parameters from a possible subset. May be useful in assessing correlation between parameter nodes, or problematic posterior surfaces.

Usage

```
pairstrace_jags(
  x,
  p = NULL,
  points = FALSE,
  contour = FALSE,
  lwd = 1,
  alpha = 0.2,
  parmfrow = NULL,
  ...
)
```

Arguments

`x`	Output object returned from `jagsUI`
`p`	Optional vector of parameters to subset
`points`	Whether to plot as scatter plots instead. Defaults to `FALSE`.
`contour`	Whether to plot as contour plots instead. Defaults to `FALSE`.
`lwd`	Line width for trace plots. Defaults to 1.
`alpha`	Opacity of lines (or points, when `points=TRUE`). Defaults to 0.2.
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`...`	additional plotting arguments or arguments to `tracedens_jags()`

Value

NULL

Author(s)

Matt Tyers

See Also

tracedens_jags, crossplot

Examples

```
pairstrace_jags(SS_out, p="sig", parmfrow=c(2,3), lwd=2)
pairstrace_jags(SS_out, p="sig", parmfrow=c(2,3), points=TRUE)
pairstrace_jags(SS_out, p="sig", parmfrow=c(2,3), contour=TRUE)

pairstrace_jags(asdf_jags_out, parmfrow=c(3,3))
pairstrace_jags(asdf_jags_out, parmfrow=c(3,3), points=TRUE)
pairstrace_jags(asdf_jags_out, parmfrow=c(3,3), contour=TRUE)
```

plotcor_jags *Plot a correlation matrix from a JAGS object*

Description

Plots a correlation matrix of all MCMC samples from an object returned by 'jagsUI', or an optional subset of parameter nodes. Correlation is plotted as shades of red (positive) or blue (negative).

In the case of vectors or arrays of nodes for each parameter name, a single axis tick will be used for all nodes with a single name. This has the effect of giving greater visual weight to single parameters, and reducing plot clutter.

Values of correlation are overlayed for all parameters with few nodes, with character size scaled according to the absolute correlation.

Usage

```
plotcor_jags(
  x,
  p = NULL,
  exact = FALSE,
  mincor = 0,
  maxn = 4,
  maxcex = 1,
  legend = TRUE,
  ...
)
```

Arguments

`x`	Output object returned from `jagsUI`, or a data.frame with MCMC output
`p`	Optional string to begin posterior names. If `NULL` is used, all parameters will be used
`exact`	Whether name must be an exact match (`TRUE`) or with initial sub-string matching only supplied characters (`FALSE`). Defaults to `FALSE`.
`mincor`	Minimum (absolute) correlation to use for text labels. Defaults to 0 (all will be plotted)
`maxn`	Maximum number of nodes per parameter name for text labels, to prevent plot clutter. Defaults to 4.
`maxcex`	Maximum character expansion factor for text labels. Defaults to 1.
`legend`	Whether to produce a plot legend. Defaults to `TRUE`.
`...`	Optional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

cor_jags

Examples

```
plotcor_jags(asdf_jags_out, maxcex=0.7)

plotcor_jags(SS_out, p=c("trend","rate","sig"))
```

`plotdens` *Plot kernel densities of single parameter nodes*

Description

Produces a kernel density plot of a single or multiple parameter nodes (overlayed).

Input can be of multiple possible formats: either a single or list of output objects from `jagsUI` with an associated vector of parameter names, or a vector or `data.frame` of posterior samples.

Usage

```
plotdens(
  df,
  p = NULL,
  exact = FALSE,
  add = FALSE,
  col = NULL,
  shade = TRUE,
  lwd = 2,
  minCI = 0.99,
  legend = TRUE,
  legendpos = "topleft",
  legendnames = NULL,
  main = NULL,
  xlab = "",
  ylab = "Density",
  ...
)
```

Arguments

`df`	Input object for plotting. See examples below.
`p`	Vector of parameter names, if `df` is given as a single or list of output objects from `jagsUI`
`exact`	Whether the `p=` argument should match the parameter name exactly. See jags_df for details.
`add`	Whether to add to an existing plot (`TRUE`) or produce a new plot. Defaults to `FALSE`.
`col`	Vector of colors for plotting. If the default (`NULL`) is accepted, colors will be automatically selected.
`shade`	Whether to shade the regions below the kernel density curve(s). Defaults to `TRUE`.
`lwd`	Line width for kernel density curves. Defaults to `2`. Note: setting this to `0` (or `FALSE`) will suppress lines.
`minCI`	Minimum CI width to include for all density curves. Defaults to 99%.
`legend`	Whether to plot a legend. Defaults to `TRUE`.
`legendpos`	Position for automatic legend. Defaults to `"topleft"`.
`legendnames`	Names for legend
`main`	Plot title. Defaults to "".
`xlab`	X-axis label. Defaults to "".
`ylab`	Y-axis label. Defaults to "Density".
`...`	Optional plotting arguments

Value

NULL

Author(s)

Matt Tyers

See Also

comparedens, comparecat, comparepriors

Examples

```
## jagsUI object with a single parameter
plotdens(asdf_jags_out, p="b1")

## jagsUI object with multiple nodes of a parameter
plotdens(asdf_jags_out, p="a")

## jagsUI object with multiple parameter nodes
plotdens(asdf_jags_out, p=c("a[1]","a[2]","a[3]"))

## data.frame with multiple columns
plotdens(jags_df(asdf_jags_out, p="a"))

## list of jagsUI objects with a single parameter name
plotdens(list(asdf_jags_out,asdf_jags_out,asdf_jags_out), p="b1")

## list of jagsUI objects with a vector of parameter names
plotdens(list(asdf_jags_out,asdf_jags_out,asdf_jags_out), p=c("a[1]","a[2]","a[3]"))
```

`plotRhats` *Plotting all Rhat values*

Description

Plotting all values of `Rhat` (or alternately `n.eff`) from an output object returned by `jagsUI`, or perhaps a subset of parameters. This function is intended as a quick graphical check of which parameters have adequately converged.

`Rhat` (Gelman-Rubin Convergence Diagnostic, or Potential Scale Reduction Factor) is calculated within 'JAGS', and is commonly used as a measure of convergence for a given parameter node. Values close to 1 are seen as evidence of adequate convergence. `n.eff` is also calculated within 'JAGS', and may be interpreted as a crude measure of effective sample size for a given parameter node.

Usage

```
plotRhats(
  x,
  p = NULL,
  n.eff = FALSE,
  fence = NULL,
  plotsequence = FALSE,
  splitarr = FALSE,
  margin = NULL,
  ...
)
```

Arguments

`x`	Output object returned from `jagsUI`
`p`	Optional vector of parameters to subset
`n.eff`	Optionally, whether to plot `n.eff` instead of `Rhat`. Defaults to `FALSE`.
`fence`	Value of horizontal lines to overlay as reference. Accepting the default value (`NULL`) will give `fence` values of 1.1 (a commonly used value) and 1.01 for Rhat, or 100 and 500 for `n.eff`.
`plotsequence`	Whether to plot parameter vectors (or matrices) in a sequence, running left to right, which may be useful for time series models, etc. If the default (`FALSE`) is used, a vertical cluster will be plotted for each parameter, resulting in a simpler plot if there are many parameters. Note that the `Rhat` values will still be plotted in sequence if the default (`FALSE`) is used.
`splitarr`	Whether to split 2+ dimensional parameter arrays by a given dimension, rather than plotting the full array in one vertical cluster or continuous sequence. This may be recommended in the case of large arrays. Defaults to `FALSE`.
`margin`	If `splitarr=` is set to `TRUE`, which array margin to split by. In the case of a 2-dimensional array, setting `margin=2` will separate the array by column. If the default (`NULL`) is accepted, the function will split by the smallest dimension, therefore splitting into the fewest groups.
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

References

Gelman, A., & Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science, 7*(4), 457–472. http://www.jstor.org/stable/2246093

See Also

traceworstRhat, check_Rhat, qq_postpred, ts_postpred, plot_postpred, kfold

Examples

```
## plotting everything
plotRhats(SS_out)
str(SS_out$Rhat)  # the associated values

plotRhats(SS_out, n.eff=TRUE)
str(SS_out$n.eff)  # the associated values

## behavior of splitarr and margin are shown
plotRhats(SS_out)
plotRhats(SS_out, splitarr=TRUE)
str(SS_out$Rhat) # the associated values

## plotsequence may be useful in the case of a sequence of values
plotRhats(SS_out, p=c("trend", "cycle_s"), splitarr=TRUE, plotsequence=TRUE)
```

`plot_postpred` *Diagnostic plots from posterior predictive distribution*

Description

This is a wrapper function that produces a sequence of plots illustrating the posterior predictive distribution. Optional plots are:

- An envelope plot of the posterior predictive distribution as a time series, overlayed with the data values (if `plot_data=TRUE` is used)
- The centered posterior predictive distributions, as plotted by ts_postpred, and overlayed with the data residuals (if `plot_residuals=TRUE` is used)
- The approximate residual standard deviation, calculated from a moving window of 10 data points in sequence. (if `plot_sd=TRUE` is used)

These three plots are repeated, for a sequence of different variables expressed on the x-axis, potentially highlighting different features of the dataset or model structure:

- The data sequence (if `whichplots=` contains `1`)
- The `x=` variable supplied (if `whichplots=` contains `2`)
- The `y=` variable supplied (if `whichplots=` contains `3`)
- The fitted values, as estimated by the posterior predictive median (if `whichplots=` contains `4`)

While not an omnibus posterior predictive check, this plot can be useful for detecting an overparameterized model, or else improper specification of observation error.

It should be noted that this function will only produce meaningful results with a vector of data, as opposed to a single value.

The posterior predictive distribution can be specified in two possible ways: either a single output object from `jagsUI` with an associated parameter name, or as a matrix or `data.frame` of posterior samples.

Usage

```
plot_postpred(
  ypp,
  y,
  p = NULL,
  x = NULL,
  whichplots = c(1, 2, 4),
  plot_data = TRUE,
  plot_residuals = TRUE,
  plot_sd = TRUE,
  pch = 1,
  pointcol = 1,
  lines = FALSE,
  ...
)
```

Arguments

`ypp`	Either a matrix or `data.frame` of posterior samples, or an output object returned from `jagsUI` and a supplied parameter name
`y`	The associated data vector
`p`	A character name, if a `jagsUI` object is passed to `ypp`
`x`	The time measurements associated with time series `y`. If the default `NULL` is accepted, associated plots will be suppressed.
`whichplots`	A vector of which sets of plots to produce (that is, with respect to which variables on the x-axis). See above for details. Defaults to `c(1, 2, 4)`.
`plot_data`	Whether to produce plots associated with the data (y=) time series and untransformed posterior predictive distribution. Defaults to `TRUE`.
`plot_residuals`	Whether to produce plots associated with the residual time series and posterior predictive residuals. Defaults to `TRUE`.
`plot_sd`	Whether to produce plots of the moving-window standard deviation of the residuals. Defaults to `TRUE`.
`pch`	Plotting character for points, which will accept a vector input. See points. Defaults to `1`.
`pointcol`	Plotting color for points. Defaults to `1`.
`lines`	Whether to add a line linking data time series points. Defaults to `FALSE`.
`...`	Additional arguments to envelope

Value

`NULL`

Note

This function assumes the existence of a matrix of posterior predictive samples corresponding to a data vector, the construction of which must be left to the user. This can be accomplished within JAGS, or using appropriate simulation from the posterior samples.

Author(s)

Matt Tyers

See Also

qq_postpred, ts_postpred, kfold, check_Rhat, check_neff, traceworstRhat, plotRhats

Examples

```
# first, a quick look at the example data...
str(SS_data)
str(SS_out$sims.list$ypp)

# recommended usage
parmfrow <- par("mfrow") # storing graphics state
par(mfcol = c(3,3))  # a recommended setting to organize plots

plot_postpred(ypp=SS_out, p="ypp", y=SS_data$y, x=SS_data$x)

par(mfrow = parmfrow) # resetting graphics state
```

`pull_post` *Subset from posterior data.frame*

Description

Extracts a subset vector or `data.frame` from a `data.frame` consisting of more columns, such that column names match a name given in the p= argument. This may be useful in creating smaller objects consisting of MCMC samples.

Usage

```
pull_post(x, p = NULL, exact = FALSE)
```

Arguments

`x`	Posterior `data.frame`
`p`	String to begin posterior names. If `NULL` is used, all parameters will be returned.
`exact`	Whether name must be an exact match (`TRUE`) or with initial sub-string matching only supplied characters (`FALSE`). Defaults to `FALSE`.

Value

A `data.frame` with a column associated with each (subsetted) parameter and a row associated with each MCMC iteration.

Author(s)

Matt Tyers

See Also

jags_df

Examples

```
out_df <- jags_df(asdf_jags_out)

b <- pull_post(out_df, p="b")
str(b)
a <- pull_post(out_df, p=c("a","sig_a"))
str(a)
sigs <- pull_post(out_df, p="sig")
str(sigs)
justsig <- pull_post(out_df, p="sig", exact=TRUE)
str(justsig)
```

`qq_postpred` *Quantile-quantile plot from posterior predictive distribution*

Description

Produces a quantile-quantile plot, calculated from the quantiles of a vector of data (most likely a time series), with respect to the matrix of associated posterior predictive distributions.

While not an omnibus posterior predictive check, this plot can be useful for detecting an overparameterized model, or else improper specification of observation error. Like a traditional Q-Q plot, a well-specified model will have points that lie close to the x=y line. In the case of this function, an overparametrized model will typically produce a plot with a much shallower slope, possibly with many associated posterior predictive quantiles close to 0.5.

It should be noted that this function will only produce meaningful results with a vector of data, as opposed to a single value.

The posterior predictive distribution can be specified in two possible ways: either a single output object from `jagsUI` with an associated parameter name, or as a matrix or `data.frame` of posterior samples.

Usage

```
qq_postpred(ypp, y, p = NULL, add = FALSE, ...)
```

Arguments

ypp	Either a matrix or `data.frame` of posterior samples, or an output object returned from `jagsUI` and a supplied parameter name
y	The associated data vector
p	A character name, if a `jagsUI` object is passed to `ypp`
add	Whether to add the plot to an existing plot. Defaults to `FALSE`.
...	Optional plotting arguments

Value

`NULL`

Note

This function assumes the existence of a matrix of posterior predictive samples corresponding to a data vector, the construction of which must be left to the user. This can be accomplished within JAGS, or using appropriate simulation from the posterior samples.

Author(s)

Matt Tyers

See Also

ts_postpred, plot_postpred, kfold, check_Rhat, check_neff, traceworstRhat, plotRhats

Examples

```
# first, a quick look at the example data...
str(SS_data)
str(SS_out$sims.list$ypp)

# plotting the example posterior predictive distribution with the data
# points overlayed.  Note the overdispersion in the posterior predictive.
caterpillar(SS_out, p="ypp")
points(SS_data$y)

# using a jagsUI object as ypp input
qq_postpred(ypp=SS_out, p="ypp", y=SS_data$y)

# using a matrix as ypp input
qq_postpred(ypp=SS_out$sims.list$ypp, y=SS_data$y)
```

`rcolors` *Random Colors*

Description

Creates a vector of randomly-generated colors.

Usage

```
rcolors(n)
```

Arguments

`n`	Vector length

Value

A vector of colors

Author(s)

Matt Tyers

Examples

```
n <- 1000
cols <- rcolors(n)
x <- runif(n)
y <- runif(n)
plot(x,y, col=cols, pch=16)
```

`skeleton` *Skeleton*

Description

Prints an example 'JAGS' model and associated 'jagsUI' code to the console, along with code to simulate a corresponding dataset. This is intended to serve as a template that can be altered as needed by the user.

Usage

```
skeleton(NAME = "NAME")
```

Arguments

`NAME`	Name to append to JAGS model object, etc.

Value

`NULL`

Note

The printed code will use the `cat()` function to write the model code to an external text file. It may be desirable to use a call to `\link{tempfile}()` instead, to eliminate creation of unneeded files.

Author(s)

Matt Tyers

Examples

```
skeleton("asdf")
```

`SS_data` *Example data: Time series associated with SS JAGS out*

Description

The time series and time measurements associated with the time series model `\link{SS_out}`.

Usage

```
SS_data
```

Format

An object of class `data.frame` with 41 rows and 2 columns.

`SS_out` *Example data: SS JAGS out*

Description

A time series model with multiple observations of a single time series, and with two stochastic cycle components.

Usage

```
SS_out
```

Format

An object of class `jagsUI` of length 24.

Details

This model is included partly to show a model with vectors or 2-dimensional matrices of parameter nodes, and also to give an example of poor model convergence.

---

tracedens_jags — *Combination of trace plots and by-chain kernel densities of* `jagsUI` *object*

Description

Combination of trace plots and by-chain kernel densities of a whole `jagsUI` object, or optional subset of parameter nodes.

Usage

```
tracedens_jags(
  x,
  p = NULL,
  exact = FALSE,
  parmfrow = NULL,
  lwd = 1,
  shade = TRUE,
  ...
)
```

Arguments

`x`	Posterior `jagsUI` object
`p`	Parameter name for subsetting: if this is specified, only parameters with names beginning with this string will be plotted.
`exact`	Whether `p` should be an exact match (`TRUE`) or just match the beginning of the string (`FALSE`). Defaults to `FALSE`.
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`lwd`	Line width for plotting. Defaults to 1.
`shade`	Whether to add semi-transparent shading to by-chain kernel densities. Defaults to `TRUE`.
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

trace_jags, chaindens_jags, pairstrace_jags

Examples

```
tracedens_jags(asdf_jags_out, parmfrow=c(4,2))
tracedens_jags(asdf_jags_out, p="a", parmfrow=c(3,1))
```

`traceworstRhat` *Trace plots corresponding to the worst values of Rhat*

Description

Trace plots with kernel densities will be created for parameters with the largest (worst) associated values of `Rhat`. This function is primarily intended for parameters with a vector (or array) of values.

`Rhat` (Gelman-Rubin Convergence Diagnostic, or Potential Scale Reduction Factor) is calculated within 'JAGS', and is commonly used as a measure of convergence for a given parameter node. Values close to 1 are seen as evidence of adequate convergence. `n.eff` is also calculated within 'JAGS', and may be interpreted as a crude measure of effective sample size for a given parameter node.

Usage

```
traceworstRhat(x, p = NULL, n.eff = FALSE, margin = NULL, parmfrow = NULL, ...)
```

Arguments

`x`	Output object returned from `jagsUI`
`p`	Optional vector of parameters to subset
`n.eff`	Whether to plot parameters with the smallest associated values of `n.eff` instead. Defaults to `FALSE`.
`margin`	In the case of a 2+ dimensional array associated with a given parameter, this will have the effect of plotting the worst `Rhat` corresponding to each margin specified. For example, specifying `margin=2` (column) will plot the parameter with the worst `Rhat` value from each column. In contrast, specifying `margin=NULL` (the default) will cause the function to plot the single array element with the largest Rhat value.
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`...`	additional plotting arguments or arguments to `tracedens_jags()`

Value

`NULL`

Author(s)

Matt Tyers

References

Gelman, A., & Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science, 7*(4), 457–472. http://www.jstor.org/stable/2246093

See Also

plotRhats, check_Rhat, qq_postpred, ts_postpred, plot_postpred, kfold

Examples

```
## plotting everything
traceworstRhat(SS_out, parmfrow=c(3,2))
SS_out$Rhat  # the associated values

traceworstRhat(SS_out, parmfrow=c(3,2), n.eff=TRUE)
SS_out$n.eff  # the associated values

## in the case of a 2-D array, setting margin=2 gives the max Rhat
## associated with each column, rather than the global max
traceworstRhat(x=SS_out, p="cycle_s", margin=2, parmfrow=c(2,2))
SS_out$Rhat
traceworstRhat(x=SS_out, p="cycle_s", margin=2, parmfrow=c(2,2), n.eff=TRUE)
SS_out$n.eff
```

---

trace_df *Trace plot of each column of a* `data.frame`.

Description

Trace plot of each column of a posterior 'data.frame'.

Usage

```
trace_df(df, nline, parmfrow = NULL, ...)
```

Arguments

`df`	Posterior data.frame
`nline`	Number of chains
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`...`	additional plotting arguments or arguments to `trace_line()`

Value

NULL

Author(s)

Matt Tyers

See Also

tracedens_jags, trace_jags, trace_line

Examples

```
a <- jags_df(asdf_jags_out, p="a")

trace_df(a, nline=3, parmfrow=c(3,1))
```

---

trace_jags *Trace plot of jagsUI object*

Description

Trace plot of a whole `jagsUI` object, or optional subset of parameter nodes.

Usage

```
trace_jags(x, p = NULL, exact = FALSE, parmfrow = NULL, lwd = 1, ...)
```

Arguments

`x`	Posterior `jagsUI` object
`p`	Parameter name for subsetting: if this is specified, only parameters with names beginning with this string will be plotted.
`exact`	Whether `p` should be an exact match (`TRUE`) or just match the beginning of the string (`FALSE`). Defaults to `FALSE`.
`parmfrow`	Optional call to `par(mfrow)` for the number of rows & columns of plot window. Returns the graphics device to previous state afterward.
`lwd`	Line width for plotting. Defaults to 1.
`...`	additional plotting arguments

Value

NULL

Author(s)

Matt Tyers

See Also

tracedens_jags, pairstrace_jags, trace_df, trace_line

Examples

```
trace_jags(asdf_jags_out, parmfrow=c(4,2))
trace_jags(asdf_jags_out, p="a", parmfrow=c(3,1))
```

trace_line *Simple trace plot*

Description

Trace plot of a single parameter node.

Usage

```
trace_line(x, nline, lwd = 1, main = "", ...)
```

Arguments

`x`	Posterior vector
`nline`	Number of MCMC chains
`lwd`	Line width
`main`	Plot title
`...`	additional plotting arguments

Value

`NULL`

Author(s)

Matt Tyers

See Also

tracedens_jags, trace_jags, trace_df, chaindens_line

Examples

```
b1 <- jags_df(asdf_jags_out, p="b1")

trace_line(b1, nline=3, main="b1")
```

ts_postpred *Time series plot of centered posterior predictive distribution*

Description

Produces a plot of centered posterior predictive distributions associated with a vector of data (most likely a time series), defined as the difference between posterior predictive and posterior predictive median.

Also overlays the posterior predictive residuals, defined as the differences between data values and their respective posterior predictive medians.

While not an omnibus posterior predictive check, this plot can be useful for detecting an overparameterized model, or else improper specification of observation error.

It should be noted that this function will only produce meaningful results with a vector of data, as opposed to a single value.

The posterior predictive distribution can be specified in two possible ways: either a single output object from `jagsUI` with an associated parameter name, or as a matrix or `data.frame` of posterior samples.

Usage

```
ts_postpred(
  ypp,
  y,
  p = NULL,
  x = NULL,
  lines = FALSE,
  pch = 1,
  pointcol = 1,
  transform = c("none", "exp", "expit"),
  ...
)
```

Arguments

`ypp`	Either a matrix or `data.frame` of posterior samples, or an output object returned from `jagsUI` and a supplied parameter name
`y`	The associated data vector
`p`	A character name, if a `jagsUI` object is passed to `ypp`
`x`	The time measurements associated with time series `y`. If the default `NULL` is accepted, equally-spaced integer values will be used.
`lines`	Whether to add a line linking data time series points. Defaults to `FALSE`.
`pch`	Plotting character for points, which will accept a vector input. See points. Defaults to `1`.
`pointcol`	Plotting color for points. Defaults to `1`.

`transform`	Should the y-axis be (back)transformed? Options are `"exp"`, indicating exponential, or `"expit"`, indicating inverse-logit. Defaults to `"none"`, indicating no transformation. Note: if `transform="exp"`is used, consider adding additional plotting argument `log="y"`.
`...`	Additional arguments to envelope

Value

`NULL`

Note

This function assumes the existence of a matrix of posterior predictive samples corresponding to a data vector, the construction of which must be left to the user. This can be accomplished within JAGS, or using appropriate simulation from the posterior samples.

Author(s)

Matt Tyers

See Also

qq_postpred, plot_postpred, kfold, check_Rhat, check_neff, traceworstRhat, plotRhats

Examples

```
# first, a quick look at the example data...
str(SS_data)
str(SS_out$sims.list$ypp)

# plotting the example posterior predictive distribution with the data
# points overlayed.  Note the overdispersion in the posterior predictive.
caterpillar(SS_out, p="ypp")
points(SS_data$y)

# using a jagsUI object as ypp input
ts_postpred(ypp=SS_out, p="ypp", y=SS_data$y)

# using a matrix as ypp input
ts_postpred(ypp=SS_out$sims.list$ypp, y=SS_data$y)

# exp transformation
ts_postpred(ypp=SS_out, p="ypp", y=SS_data$y, transform="exp")
ts_postpred(ypp=SS_out, p="ypp", y=SS_data$y, transform="exp", log="y")
```

Index