Package 'Boom'

September 3, 2019

Version 0.9.2

Date 2019-09-01

Title Bayesian Object Oriented Modeling

Author Steven L. Scott is the sole author and creator of the BOOM project. Some code in the BOOM libraries has been modified from other open source projects. These include Cephes (obtained from Netlib, written by Stephen L. Moshier), NEWUOA (M.J.D Powell, obtained from Powell's web site), and a modified version of the R math libraries (R core development team). Original copyright notices have been maintained in all source files. In these cases, copyright claimed by Steven L. Scott is limited to modifications made to the original code. Google claims copyright for code written while Steven L. Scott was employed at Google from 2008 - 2018, but BOOM is not an officially supported Google project.

Maintainer Steven L. Scott <steve.the.bayesian@gmail.com>

Description A C++ library for Bayesian modeling, with an emphasis on Markov chain Monte Carlo. Although boom contains a few R utilities (mainly plotting functions), its primary purpose is to install the BOOM C++ library on your system so that other packages can link against it.

License LGPL-2.1 | file LICENSE

Depends MASS, R(>= 3.1.0)

Suggests testthat

Encoding UTF-8

SystemRequirements GNU Make, C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-09-03 21:00:09 UTC

R topics documented:

`add.segments` *Function to add horizontal line segments to an existing plot*

Description

Adds horizontal line segments to an existing plot. The segments are centered at `x` with height `y`. The `x` values are assumed to be equally spaced, so that `diff(x)` is a constant 'dx'. The line segments go from `x +/-half.width.factor *dx`, so if `half.width.factor=.5` there will be no gaps between segments. The default is to leave a small gap.

This function was originally used to add reference lines to side-by-side boxplots.

Usage

```
AddSegments(x, y, half.width.factor = 0.45, ...)
```

Arguments

`x`	A numeric vector giving the midpoints of the line segments.
`y`	A numeric vector of the same length as `x` giving the vertical position of the line segments
`half.width.factor`	See 'description' above.
`...`	graphical parameters controlling the type of lines used in the line segments

Value

Called for its side effect.

Author(s)

Steven L. Scott

See Also

`boxplot.true`

Examples

```
x <- rnorm(100)
y <- rnorm(100, 1)
boxplot(list(x=x,y=y))
AddSegments(1:2, c(0, 1))  ## add segments to the boxplot
```

---

ar1.coefficient.prior *Normal prior for an AR1 coefficient*

Description

A (possibly truncated) Gaussian prior on the autoregression coefficient in an AR1 model.

Usage

```
Ar1CoefficientPrior(mu = 0, sigma = 1, force.stationary = TRUE,
    force.positive = FALSE, initial.value = mu)
```

Arguments

`mu`	The mean of the prior distribution.
`sigma`	The standard deviation of the prior distribution.
`force.stationary`	Logical. If `TRUE` then the prior support for the AR1 coefficient will be truncated to (-1, 1).
`force.positive`	Logical. If `TRUE` then the prior for the AR1 coefficient will be truncated so that zero support is given to values less than zero.
`initial.value`	The initial value of the parameter being modeled in the MCMC algorithm.

Details

The `Ar1CoefficientPrior()` syntax is preferred, as it more closely matches R's syntax for other constructors.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

beta.prior *Beta prior for a binomial proportion*

Description

Specifies beta prior distribution for a binomial probability parameter.

Usage

```
BetaPrior(a = 1, b = 1, mean = NULL, sample.size = NULL,
          initial.value = NULL)
```

Arguments

a	A positive real number interpretable as a prior success count.
b	A positive real number interpretable as a prior failure count.
mean	A positive real number representing a/(a+b).
sample.size	A positive real number representing a+b.
initial.value	An initial value to be used for the variable being modeled. If NULL then the mean of the distribution will be used instead.

Details

The distribution should be specified either with a and b, or with mean and sample.size.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

boxplot.mcmc.matrix *Plot the distribution of a matrix*

Description

Plot the marginal distribution of each element in the Monte Carlo distribution of a matrix (e.g. a variance matrix or transition probability matrix). Rows and columns in the boxplots correspond to rows and columns in the matrix being plotted.

Usage

```
BoxplotMcmcMatrix(X, ylim = range(X), col.names,
                  row.names, truth, colors = NULL,
                  las = 0, ...)
```

Arguments

X	3 dimensional array. The first dimension is the Monte Carlo index (e.g. MCMC iteration). The second and third dimensions are the row and column of the matrix being plotted. E.g. X[i,j,k] is Monte Carlo draw i of matrix element j,k.
ylim	2-vector giving the lower and upper limits of the vertical axis.
col.names	(optional) character vector giving the names of matrix columns (third dimension of X).
row.names	(optional) character vector giving the names of matrix rows (second dimension of X).
truth	(optional) scalar or matrix giving the values of reference lines to be plotted on each boxplot. If a scalar then the same value will be used for each boxplot. If a matrix then the rows and columns of the matrix correspond to the second and third dimension of X.
colors	A vector of colors to use for the boxplots. Each row uses the same color scheme.
las	Controls the orientation of axis labels. See the las section in the help page for par.
...	Extra arguments passed to boxplot

Value

Called for its side effect, which is to draw a set of side-by-side boxplots on the current graphics device.

Author(s)

Steven L. Scott

See Also

boxplot.true, boxplot

Examples

```
X <- array(rnorm(1000 * 3 * 4), dim=c(1000, 3, 4))
dimnames(X)[[2]] <- paste("row", 1:3)
dimnames(X)[[3]] <- paste("col", 1:4)
BoxplotMcmcMatrix(X)

truth <- 0
BoxplotMcmcMatrix(X, truth=truth)

truth <- matrix(rnorm(12), ncol=4)
BoxplotMcmcMatrix(X, truth=truth)
```

boxplot.true *Compare Boxplots to True Values*

Description

Plots side-by-side boxplots of the columns of the matrix x. Each boxplot can have its own reference line (truth) and standard error lines se.truth, if desired. This function was originally written to display MCMC output, where the reference lines were true values used to test an MCMC simulation.

Usage

```
BoxplotTrue(x, truth = NULL, vnames = NULL, center = FALSE,
            se.truth = NULL, color = "white", truth.color = "black",
            ylim = NULL, ...)
```

Arguments

x	The matrix whose columns are to be plotted.
truth	(optional) A vector of reference values with length equal to ncol(x).
vnames	(optional) character vector giving the column names of x.
center	(optional) logical. If truth is supplied then center=TRUE will center each column of x around truth to show the variation around the reference line.
se.truth	(optional) numeric vector of length ncol(x). If truth is supplied then additional reference lines will be drawn at truth +/-2*se.truth.
color	(optional) vector of colors for each boxplot.
truth.color	A color (or vector of colors) to use for the segments representing true values.
ylim	Limits for the vertical axis. If NULL then the axis will be scaled to fit x, truth, and truth + c(2,-2) * se.truth
...	additional arguments to boxplot.

Value

called for its side effect

Author(s)

Steven L. Scott

See Also

boxplot.matrix, boxplot,

Examples

```
x <- t(matrix(rnorm(5000, 1:5, 1:5), nrow=5))
BoxplotTrue(x, truth=1:5, se.truth=1:5, col=rainbow(5), vnames =
  c("EJ", "TK", "JT", "OtherEJ", "TJ") )
```

check *Check MCMC Output*

Description

Verify that MCMC output covers expected values.

Usage

```
CheckMcmcMatrix(draws, truth, confidence = .95,
                control.multiple.comparisons = TRUE,
                burn = 0)

CheckMcmcVector(draws, truth, confidence = .95, burn = 0)

McmcMatrixReport(draws, truth, confidence = .95, burn = 0)
```

Arguments

draws	The array of MCMC draws to check. This must be a matrix for CheckMcmcMatrix and a vector for CheckMcmcVector.
truth	The vector of true values that must be covered by draws in order for the check to succeed.
confidence	Specifies the probability width of the intervals used to determine whether draws covers truth. Central intervals are used, not HPD intervals.
control.multiple.comparisons	If FALSE then every interval must cover its corresponding true value. Otherwise a fraction of intervals (given by confidence) must cover.
burn	The number of MCMC iterations to discard as burn-in.

Details

CheckMcmcVector checks a vector of draws corresponding to a scalar random variable. CheckMcmcMatrix checks a matrix of draws corresponding to a vector of random variables. In either case the check is made by constructing a central confidence interval (obtained by removing half of 1 -confidence from the upper and lower tails of the distribution).

If a single variable is being checked with CheckMcmcVector then the check passes if and only if the interval covers the true value.

If multiple values are being checked with CheckMcmcMatrix then the user has control over how strict to make the check. If control.multiple.comparisons is FALSE then the check passes if and only if all intervals cover true values. Otherwise a fraction of intervals must cover. The fraction is the lower bound of the binomial confidence interval for the coverage rate under the hypothesis that the true coverage rate is confidence.

Value

CheckMcmcVector and CheckMcmcMatrix return TRUE if the check passes, and FALSE if it does not.

McmcMatrixReport returns a string that can be put in the info field of an expect_true expression, to give useful information about a failed test case. The return value is a textual representation of a three column matrix. Each row matches a variable in draws, and gives the lower and upper bounds for the credible interval used to check the values. The final column lists the true values that are supposed to be inside the credible intervals. The value is returned as a character string that is expected to be fed to cat() or print() so that it will render correctly in R CMD CHECK output.

Author(s)

Steven L. Scott

Examples

```
ndraws <- 100
draws <- rnorm(ndraws, 0, 1)
CheckMcmcVector(draws, 0)  ## Returns TRUE
CheckMcmcVector(draws, 17)  ## Returns FALSE

draws <- matrix(nrow = ndraws, ncol = 5)
for (i in 1:5) {
  draws[, i] <- rnorm(ndraws, i, 1)
}
CheckMcmcMatrix(draws, truth = 1:5)  ## Returns TRUE
CheckMcmcMatrix(draws, truth = 5:1)  ## Returns FALSE
```

---

check.data *Checking data formats*

Description

Checks that data matches a concept

Usage

```
check.scalar.probability(x)
check.positive.scalar(x)
check.nonnegative.scalar(x)
check.probability.distribution(x)
check.scalar.integer(x)
check.scalar.boolean(x)
```

Arguments

x	An object to be checked.

Details

If the object does not match the concept being checked, stop is called. Otherwise TRUE is returned.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

---

circles *Draw Circles*

Description

Draw circles on the current graphics device.

Usage

```
circles(center, radius, ...)
```

Arguments

center	A two-column matrix giving the coordinates of the circle center. If a single circle is to be drawn then a 2-element vector can be passed instead.
radius	The radii of the circles. A scalar value will be repeated if center is a matrix with more than one row.
...	Extra arguments passed to 'segments'. See par for options controlling line type, line width, color, etc.

Details

Draws circles on the current graphics device. This is a low-level plotting function similar to points, lines, segments, etc.

Value

Returns invisible NULL.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
plot(1:10, type = "n")
circles(cbind(c(2, 3, 4), c(4, 5,6 )), radius = c(.3, .4, .5))
```

compare.den *Compare several density estimates.*

Description

Produces multiple density plots on a single axis, to compare the columns of a matrix or the elements of a list.

Usage

```
CompareDensities(x,
                 legend.text = NULL,
                 legend.location = "topright",
                 legend.title = NULL,
                 xlim = NULL,
                 ylim = NULL,
                 xlab = "parameter",
                 ylab = "density",
                 main = "",
                 lty = NULL,
                 col = "black",
                 axes = TRUE,
                 na.rm = TRUE,
                 ...)
```

Arguments

x	matrix or list of numeric vectors. A density plot is produced for each column of the matrix or element of the list.
legend.text	(optional) character vector giving names of each density plot.
legend.location	Entry that can be passed to legend.
legend.title	The legend title.
xlim	(optional) horizonal range of the plotting region. If omitted the region will be sized to fit all the observations in x.
ylim	(optional) vertical range of the plotting region. If omitted the region will be sized to fit all empirical density plots.
xlab	label to be placed on the horizontal axis
ylab	label to be placed on the vertical axis
main	main title for the plot
lty	The line types to use for the different densities. See par. If NULL then a different line type will be used for each density.
col	vector of colors for the densities to be plotted.
axes	Logical. Should axes and a box be drawn around the figure?

na.rm	Logical value indicating whether NA's should be removed.
...	Other graphical parameters passed to plot.density, and lines.

Value

Called for its side effect, which is to produce multiple density plots on the current graphics device.

Author(s)

Steven L. Scott

See Also

density

Examples

```
x <- t(matrix(rnorm(5000, 1:5, 1:5), nrow=5))
CompareDensities(x, legend.text=c("EJ", "TK", "JT", "OtherEJ", "TJ"),
                 col=rainbow(5), lwd=2)
```

---

compare.dynamic.distributions

Compare Dynamic Distributions

Description

Produce a plot showing several stacked dynamic distributions over the same horizontal axis.

Usage

```
CompareDynamicDistributions(
    list.of.curves,
    timestamps,
    style = c("dynamic", "boxplot"),
    xlab = "Time",
    ylab = "",
    frame.labels = rep("", length(list.of.curves)),
    main = "",
    actuals = NULL,
    col.actuals = "blue",
    pch.actuals = 1,
    cex.actuals = 1,
    vertical.cuts = NULL,
    ...)
```

Arguments

list.of.curves	A list of matrices, all having the same number of columns. Each matrix represents a distribution of curves, with rows corresponding to individual curves, and columns to time points.
timestamps	A vector of time stamps, with length matching the number of columns in each element of list.of.curves.
style	Should the curves be represented using a dynamic distribution plot, or boxplots. Boxplots are better for small numbers of time points. Dynamic distribution plots are better for large numbers of time points.
xlab	Label for the horizontal axis.
ylab	Label for the (outer) vertical axis.
frame.labels	Labels for the vertical axis of each subplot. The length must match the number of plot.
main	Main title for the plot.
actuals	If non-NULL, actuals should be a numeric vector giving the actual "true" value at each time point.
col.actuals	Color to use for the actuals. See par.
pch.actuals	Plotting character(s) to use for the actuals. See par.
cex.actuals	Scale factor for actuals. See par.
vertical.cuts	If non-NULL then this must be a vector of the same type as timestamps with length matching the number of plots. A vertical line will be drawn at this location for each plot. Entries with the value NA signal that no vertical line should be drawn for that entry.
...	Extra arguments passed to PlotDynamicDistribution or TimeSeriesBoxplot.

Author(s)

Steven L. Scott

---

compare.many.densities

Compare several density estimates.

Description

Produce a plot that compares the kernel density estimates for each element in a series of Monte Carlo draws of a vector or matrix.

Usage

```
CompareManyDensities(list.of.arrays,
                     style = c("density", "box"),
                     main = "",
                     color = NULL,
                     gap = 0,
                     burn = 0,
                     suppress.labels = FALSE,
                     x.same.scale = TRUE,
                     y.same.scale = FALSE,
                     xlim = NULL,
                     ylim = NULL,
                     legend.location = c("top", "right"),
                     legend.cex = 1,
                     reflines = NULL,
                     ...)
```

Arguments

`list.of.arrays`	A list of arrays representing the MCMC draws of the vector or matrix in question. Each list element represents a different group. The first index in each list list element represents the Monte Carlo draw number (or iteration). The remaining indices represent the variables to be plotted. If the first list element has variable names assigned to its indices, these will be used to label the plots.
`style`	The style of plot to use for comparing distributions.
`main`	The main title of the plot.
`color`	A vector of colors to be used for representing the groups.
`gap`	The gap (in lines) between plots.
`burn`	The number of MCMC iterations to be discarded as burn-in.
`suppress.labels`	Logical. If `FALSE` then the dimnames (if any) of the first element in `list.of.arrays` will be used to annotate the plot. If `TRUE` then no labels will be used.
`x.same.scale`	Logical indicating whether the same horizontal scale should be used for all the plots.
`y.same.scale`	Logical indicating whether the same vertical scale should be used for all the plots. This argument is ignored if `style == "box"`.
`xlim`	Either `NULL`, or a pair of numbers giving limits for the horizontal axis. If `xlim` is set then the same xlim values will be used for all plots and the `x.same.scale` argument will be ignored.
`ylim`	Either `NULL`, or a pair of numbers giving limits for the vertical axis. If `ylim` is set then the same ylim values will be used for all plots and the `y.same.scale` argument will be ignored. This argument is ignored if `style == "box"`.
`legend.location`	The location of the legend, either on top or at the right. It can also be `NULL` in which case no legend will appear. The legend names will be taken from names(list.of.arrays). If it does not have names, then no legend will be produced.

legend.cex	The relative scale factor to use for the legend text.
reflines	This can be NULL, in which case no reference lines are drawn, it can be a single real number in which case a reference line will be drawn at that value in each panel, or it can be a vector with length equal to the number of panels, in which case a reference line will be drawn at each panel-specific value.
...	Extra arguments passed to CompareDen.

Author(s)

Steven L. Scott

See Also

density, CompareManyTs

Examples

```
x <- array(rnorm(9000), dim = c(1000, 3, 3))
dimnames(x) <- list(NULL, c("Larry", "Moe", "Curly"), c("Larry", "Eric", "Sergey"))
y <- array(rnorm(9000), dim = c(1000, 3, 3))
z <- array(rnorm(9000), dim = c(1000, 3, 3))
data <- list(x = x, y = y, z = z)
CompareManyDensities(data, color = c("red", "blue", "green"))
CompareManyDensities(data, style = "box")

x <- matrix(rnorm(5000), nrow = 1000)
colnames(x) <- c("Larry", "Moe", "Curly", "Shemp", "???")
y <- matrix(rnorm(5000), nrow = 1000)
z <- matrix(rnorm(5000), nrow = 1000)
data <- list(x = x, y = y, z = z)
CompareManyDensities(data, color = c("red", "blue", "green"))
CompareManyDensities(data, style = "box")
```

---

compare.many.ts *Compares several density estimates.*

Description

Produce a plot that compares the kernel density estimates for each element in a series of Monte Carlo draws of a vector or matrix.

Usage

```
CompareManyTs(list.of.ts, burn = 0, type = "l", gap = 0,
              boxes = TRUE, thin = 1, labels = NULL,
              same.scale = TRUE, ylim = NULL, refline = NULL,
              color = NULL, ...)
```

Arguments

list.of.ts	A list of time series matrices, data.frames or 3-dimensional arrays, all of the same size. The list elements correspond to groups. The first index of the array in each list element corresponds to time. The subsequent indices correspond to variables to be plotted.
burn	The number of initial observations to be discarded as burn-in (when plotting MCMC output).
type	The plotting type to use when plotting the time series. See plot.
gap	The amount of space to put between plots.
boxes	Logical. Should boxes be drawn around the plots?
thin	Plot every thin'th observation. This can reduce the amount of time it takes to make the plot if there are many long time series.
labels	A character vector to use as labels for individual plots.
same.scale	Logical. If TRUE then all plots are shown on the same verical scale, and vertical axes are drawn. If FALSE then each plot gets its own scale.
ylim	The scale of the vertical axis. If non-NULL then same.scale will be set to TRUE.
refline	The scalar value at which a thin dotted horizontal line should be plotted in each panel. This is useful for highlighting zero, for example.
color	A vector of colors to use for the plots.
...	Extra arguments passed to 'plot' and 'axis'.

Author(s)

Steven L. Scott

See Also

PlotManyTs, CompareManyDensities

Examples

```
x <- array(rnorm(9000), dim = c(1000, 3, 3))
dimnames(x) <- list(NULL, c("Larry", "Moe", "Curly"), c("Larry", "Eric", "Sergey"))
y <- array(rnorm(9000), dim = c(1000, 3, 3))
z <- array(rnorm(9000), dim = c(1000, 3, 3))
data <- list(x = x, y = y, z = z)
CompareManyTs(data, color = c("red", "blue", "green"))

x <- matrix(rnorm(5000), nrow = 1000)
colnames(x) <- c("Larry", "Moe", "Curly", "Shemp", "???")
y <- matrix(rnorm(5000), nrow = 1000)
z <- matrix(rnorm(5000), nrow = 1000)
data <- list(x = x, y = y, z = z)
CompareManyTs(data, color = c("red", "blue", "green"))
```

compare.vector.distribution *Boxplots to compare distributions of vectors*

Description

Uses boxplots to compare distributions of vectors.

Usage

```
CompareVectorBoxplots(draws, main = NULL, colors = NULL, burn = 0, ...)
```

Arguments

draws	A list of MCMC draws. Each list element is a matrix with rows corresponding to MCMC iterations and columns to variables. The matrices can have different numbers of rows, but should have the same numbers of columns.
main	Main title of the plot.
colors	Colors to use for the boxplots. The length must match the number entries in draws.
burn	The number of initial MCMC iterations to discard before making the plot.
...	Extra arguments passed to boxplot.

Details

Creates side-by-side boxplots with the dimensions of each vector gropued together.

Examples

```
x <- matrix(rnorm(300, mean = 1:3, sd = .4), ncol = 3, byrow = TRUE)
y <- matrix(rnorm(600, mean = 3:1, sd = .2), ncol = 3, byrow = TRUE)
CompareVectorBoxplots(list(x = x, y = y), colors = c("red", "blue"))
```

diff.double.model *DiffDoubleModel*

Description

A 'DiffDoubleModel' is tag given to a probability distribution that measures a real-valued scalar outcome, and whose log density is twice differentiable with respect to the random variable. The tag is a signal to underlying C++ code that the object being passed is one of a subset of understood distributions. Presently that subset includes the following distributions.

- SdPrior
- NormalPrior
- BetaPrior
- UniformPrior
- GammaPrior
- TruncatedGammaPrior
- LognormalPrior

Clearly this list is non-exhaustive, and other distributions may be added in the future.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

`dirichlet-distribution`

The Dirichlet Distribution

Description

Density and random generation for the Dirichlet distribution.

Usage

```
ddirichlet(probabilities, nu, logscale = FALSE)
rdirichlet(n, nu)
```

Arguments

probabilities	A vector representing a discrete probability distribution, or a matrix where each row is a discrete probability distribution. Zero probabilities are not allowed.
nu	The parameters of the Dirichlet distribution. This can be a vector of positive numbers, interpretable as prior counts, of length matching the dimension of probabilities. If probabilities is a matrix (or if n > 1) then nu can also be a matrix of the same dimension, in which case each row of nu is used to evaluate the corresponding row of probabilities.
logscale	Logical. If TRUE then return the density on the log scale. Otherwise return the density on the raw scale.
n	The number of desired draws.

Details

The Dirichlet distribution is a generalization of the beta distribution. Whereas beta distribution is a model for probabilities, the Dirichlet distribution is a model for discrete distributions with several possible outcome values.

Let π denote a discrete probability distribution (a vector of positive numbers summing to 1), and let ν be a vector of positive numbers (the parameters of the Dirichlet distribution), which can be thought of as prior counts. Then the density of the Dirichlet distribution can be written

$$f(\pi) = \frac{\Gamma(\sum_i \nu_i)}{\prod_i \Gamma(\nu_i)} \prod_i \pi_i^{\nu_i - 1}.$$

Value

`ddirichlet` returns a vector of density values, with one entry per row in `probabilities`. `rdirichlet` returns a matrix (if `n > 1`) or a vector (if `n==1`) containing the draws from the Dirichlet distribution with the specified parameters.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

---

dirichlet.prior *Dirichlet prior for a multinomial distribution*

Description

Specifies Dirichlet prior for a discrete probability distribution.

Usage

```
DirichletPrior(prior.counts, initial.value = NULL)
```

Arguments

`prior.counts`	A vector of positive numbers representing prior counts.
`initial.value`	The initial value in the MCMC algorithm of the distribution being modeled.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

discrete-uniform-prior

Discrete prior distributions

Description

Prior distributions over a discrete quantities.

Usage

```
PointMassPrior(location)
PoissonPrior(mean, lower.limit = 0, upper.limit = Inf)
DiscreteUniformPrior(lower.limit, upper.limit)
```

Arguments

location	The location of the point mass.
mean	The mean of the Poisson distribution.
lower.limit	The smallest value within the support of the distribution. The prior probability for numbers less than lower.limit is zero.
upper.limit	The largest value within the support of the distribution. The prior probability for numbers greater than upper.limit is zero.

Value

Each function returns a prior object whose class is the same as the function name. All of these inherit from "DiscreteUniformPrior" and from "Prior".

The PoissonPrior assumes a potentially truncated Poisson distribution with the given mean.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
## Specify an exact number of trees in a Bart model (see the BoomBart
## package).

ntrees <- PointMassPrior(200)

## Uniform prior between 50 and 100 trees, including the endpoints.
ntrees <- DiscreteUniformPrior(50, 100)

## Truncated Poisson prior, with a mean of 20, a lower endpoint of 1,
## and an upper endpoint of 50.
ntrees <- PoissonPrior(20, 1, 50)
```

dmvn *Multivariate Normal Density*

Description

Evaluate the multivariate normal density.

Usage

```
dmvn(y, mu, sigma, siginv = NULL, ldsi = NULL, logscale = FALSE)
```

Arguments

y	A numeric vector or matrix containing the data whose density is desired.
mu	The mean of the distribution. A vector.
sigma	The variance matrix of the distribution. A matrix.
siginv	The inverse of sigma, or NULL. If siginv is non-NULL then sigma will not be used.
ldsi	The log determinant of siginv or NULL.
logscale	Logical. If TRUE then the density is returned on the log scale. Otherwise the density is returned on the density scale.

Value

A vector containing the density of each row of y.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

double.model *Prior distributions for a real valued scalar*

Description

A DoubleModel is a class of prior distributions for real valued scalar parameters. A DoubleModel is sometimes used by a probability model that does not have a conjugate prior.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

See Also

BetaPrior, GammaPrior, LognormalPrior, NormalPrior, SdPrior, TruncatedGammaPrior, and UniformPrior.

external.legend *Add an external legend to an array of plots.*

Description

ExternalLegendLayout sets up a plotting region to plot a regular grid of points, with an optional legend on the top or right of the grid.

AddExternalLegend adds a legend to a grid of plots that was set up using ExternalLegendLayout.

Usage

```
ExternalLegendLayout(nrow,
                     ncol,
                     legend.labels,
                     legend.location = c("top", "right"),
                     outer.margin.lines = rep(4, 4),
                     gap.between.plots = rep(0, 4),
                     legend.cex = 1,
                     x.axis = TRUE,
                     y.axis = TRUE)

AddExternalLegend(legend.labels,
                  legend.location = c("top", "right"),
                  legend.cex =1,
                  bty = "n",
                  ...)
```

Arguments

nrow	The number of rows in the array of plots.
ncol	The number of columns in the array of plots.
legend.labels	The labels to be used in the legend.
legend.location	Specifies whether the legend should appear on the top or the right hand side. It can also be NULL, indicating that no legend is desired.
outer.margin.lines	A vector of length four giving the number of lines of text desired for the outer margins of the plot. See the oma argument of par. This can also be specified as a single number, to be repeated 4 times.
gap.between.plots	A vector of length 4 giving the number of lines of text to leave between grid panels. See the mar argument of par. This can also be specified as a single number, to be repeated 4 times.
legend.cex	The scale factor that will be used for legend text. This must match the scale factor used in add.external.legend.

x.axis	Will any plots have a horizontal axis?
y.axis	Will any plots have a vertical axis?
bty	Type of box to draw around the legend. Can be "n" (for no box) or "o" for a box. See legend.
...	Extra arguments passed to legend.

Value

ExternalLegendLayout returns the original graphical parameters, intended for use with on.exit.

AddExternalLegend returns invisible NULL.

Author(s)

Steven L. Scott

See Also

legend layout

Examples

```
example.plot <- function() {
  x <- rnorm(100)
  y <- rnorm(100)
  scale <- range(x, y)
  opar <- ExternalLegendLayout(nrow = 2,
                               ncol = 2,
                               legend.labels = c("foo", "bar"))
  on.exit({par(opar); layout(1)})
  hist(x, xlim = scale, axes = FALSE, main = "")
  mtext("X", side = 3, line = 1)
  box()
  plot(x, y, xlim = scale, ylim = scale, axes = FALSE)
  box()
  axis(3)
  axis(4)
  plot(y, x, xlim = scale, ylim = scale, axes = FALSE, pch = 2, col = 2)
  box()
  axis(1)
  axis(2)
  hist(y, xlim = scale, axes = FALSE, main = "")
  mtext("Y", side = 1, line = 1)
  box()
  AddExternalLegend(legend.labels = c("foo", "bar"),
                    pch = 1:2,
                    col = 1:2,
                    legend.cex = 1.5)
}

## Now call example.plot().
```

```
example.plot()

## Because of the call to on.exit(), in example.plot,
## the original plot layout is restored.
hist(1:10)
```

`gamma.prior` *Gamma prior distribution*

Description

Specifies gamma prior distribution.

Usage

```
GammaPrior(a = NULL, b = NULL, prior.mean = NULL, initial.value = NULL)
TruncatedGammaPrior(a = NULL, b = NULL, prior.mean = NULL,
                    initial.value = NULL,
                    lower.truncation.point = 0,
                    upper.truncation.point = Inf)
```

Arguments

`a`	The shape parameter in the Gamma(a, b) distribution.
`b`	The scale paramter in the Gamma(a, b) distribution.
`prior.mean`	The mean the Gamma(a, b) distribution, which is a/b.
`initial.value`	The initial value in the MCMC algorithm of the variable being modeled.
`lower.truncation.point`	The lower limit of support for the truncated gamma distribution.
`upper.truncation.point`	The upper limit of support for the truncated gamma distribution.

Details

The mean of the Gamma(a, b) distribution is a/b and the variance is a/b^2. If `prior.mean` is not `NULL`, then one of either `a` or `b` must be non-`NULL` as well.

GammaPrior is the conjugate prior for a Poisson mean or an exponential rate. For a Poisson mean `a` corresponds to a prior sum of observations and `b` to a prior number of observations. For an exponential rate the roles are reversed `a` represents a number of observations and `b` the sum of the observed durations. The gamma distribution is a generally useful for parameters that must be positive.

The gamma distribution is the conjugate prior for the reciprocal of a Guassian variance, but `SdPrior` should usually be used in that case.

A TruncatedGammaPrior is a GammaPrior with support truncated to the interval (`lower.truncation.point`,`upper.trunca`

If an object specifically needs a `GammaPrior` you typically cannot pass a `TruncatedGammaPrior`.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

GenerateFactorData *Generate a data frame of all factor data*

Description

This function is mainly intended for example code and unit testing. It generates a `data.frame` containing all factor data.

Usage

```
GenerateFactorData(factor.levels.list, sample.size)
```

Arguments

`factor.levels.list`	A list of character vectors giving factor level names. The names attribute of this list becomes the set of variables names for the return data frame.
`sample.size`	The desired number of rows in the returned data frame.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
foo <- GenerateFactorData(list(a = c("foo", "bar", "baz"),
                               b = c("larry", "moe", "curly", "shemp")),
                          50)

head(foo)
#     a     b
# 1 bar curly
# 2 foo curly
# 3 bar   moe
# 4 bar   moe
# 5 baz curly
# 6 bar curly
```

histabunch *A Bunch of Histograms*

Description

Plot a bunch of histograms describing the marginal distributions the columns in a data frame.

Usage

```
histabunch(x, gap = 1, same.scale = FALSE, boxes = FALSE,
           min.continuous = 12, max.factor = 40,
           vertical.axes = FALSE, ...)
```

Arguments

x	A matrix or data frame containing the variables to be plotted.
gap	The gap between the plots, measured in lines of text.
same.scale	Logical. Indicates whether the histograms should all be plotted on the same scale.
boxes	Logical. Indicates whether boxes should be drawn around the histograms.
min.continuous	Numeric variables with more than min.continuous unique values will be plotted as continuous. Otherwise they will be treated as factors.
max.factor	Factors with more than max.factor levels will be beautified (ha!) by combining their remaining levels into an "other" category.
vertical.axes	Logical value indicating whether the histograms should be given vertical "Y" axes.
...	Extra arguments passed to hist (for numeric variables) or barplot (for factors).

Value

Called for its side effect, which is to produce multiple histograms on the current graphics device.

Author(s)

Steven L. Scott

See Also

hist barplot

Examples

```
data(airquality)
histabunch(airquality)
```

`inverse-wishart` *Inverse Wishart Distribution*

Description

Density for the inverse Wishart distribution.

Usage

```
dInverseWishart(Sigma, sum.of.squares, nu, logscale = FALSE,
                log.det.sumsq = log(det(sum.of.squares)))

InverseWishartPrior(variance.guess, variance.guess.weight)
```

Arguments

`Sigma`	Argument (random variable) for the inverse Wishart distribution. A positive definite matrix.
`nu`	The "degrees of freedom" parameter of the inverse Wishart distribution. The distribution is only defined for `nu >= nrow(Sigma) -1`.
`sum.of.squares`	A positive definite matrix. Typically this is the sum of squares that is the sufficient statistic for the inverse Wishart distribution.
`logscale`	Logical. If `TRUE` then the density is returned on the log scale. Otherwise the density is returned on the density scale.
`log.det.sumsq`	The log determinant of `sum.of.squares`. If this function is to be called many times then precomputing the log determinant can save considerable compute time.
`variance.guess`	A prior guess at the value of the variance matrix the prior is modeling.
`variance.guess.weight`	A positive scalar indicating the number of observations worth of weight to place on `variance.guess`.

Details

The inverse Wishart distribution has density function

$$\frac{|Sigma|^{-\frac{\nu+p+1}{2}} \exp(-tr(\Sigma^{-1}S)/2)}{2^{\frac{\nu p}{2}}|\Sigma|^{\frac{\nu}{2}}\Gamma_p(\nu/2)}$$

Value

`dInverseWishart` returns the scalar density (or log density) at the specified value. This function is not vectorized, so only one random variable (matrix) can be evaluated at a time.

`InverseWishartPrior` returns a list that encodes the parameters of the distribution in a format expected by underlying C++ code.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

See Also

dWishart, rWishart, NormalInverseWishartPrior

---

invgamma *Inverse Gamma Distribution*

Description

Density, distribution function, quantile function, and random draws from the inverse gamma distribution.

Usage

```
dinvgamma(x, shape, rate, logscale = FALSE)
pinvgamma(x, shape, rate, lower.tail = TRUE, logscale = FALSE)
qinvgamma(p, shape, rate, lower.tail = TRUE, logscale = FALSE)
rinvgamma(n, shape, rate)
```

Arguments

x	A vector of deviates where the density or distribution function is to be evaluated.
p	A vector of probabilities representing CDF values (if lower.tail == TRUE) or survivor function values (if lower.tail == FALSE) from the inverse gamma distribution.
n	The desired number of draws from the inverse gamma distribution.
shape	The shape parameter.
rate	The 'rate' parameter. NOTE: The term 'rate' is used to match the corresponding parameter in rgamma. Much of the rest of the world calls this the 'scale' parameter.
logscale	Logical. If TRUE then probabilities or density values are interpreted on the log scale. Otherwise the scale is the probability or probability density scale.
lower.tail	Logical. If TRUE then cumulative probabilities are measured from zero, as in a CDF. If FALSE then cumulative are measured from infinity, as in a survivor function.

Value

- rinvgamma returns draws from the distribution.
- dinvgamma returns the density function.
- pinvgamma returns the cumulative distribution function (or survivor function, if lower.tail == FALSE).
- qinvgamma returns quantiles from the distribution. qinvgamma and pinvgamma are inverse functions.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

---

is.even *Check whether a number is even or odd.*

Description

Check whether a number is even or odd.

Usage

```
IsEven(x)
IsOdd(x)
```

Arguments

x	An integer or vector of integers.

Value

Logical indicating whether the argument is even (or odd).

Author(s)

Steven L. Scott

Examples

```
IsEven(2) ## TRUE
IsOdd(2)  ## FALSE
```

---

lmgamma *Log Multivariate Gamma Function*

Description

Returns the log of the multivariate gamma function.

Usage

```
lmgamma(y, dimension)
```

Arguments

y	The function argument, which must be a positive scalar.
dimension	The dimension of the multivariate gamma function, which must be an integer >= 1.

Details

The multivariate gamma function is

$$\Gamma_p(y) = \pi^{p(p-1)/4} \prod_{j=1}^{p} \Gamma(y + (1-j)/2).$$

The multivariate gamma function shows up as part of the normalizing constant for the Wishart and inverse Wishart distributions.

Value

Returns the log of the multivariate gamma function. Note that this function is not vectorized. Both y and dimension must be scalars, and the return value is a scalar.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

---

log.integrated.gaussian.likelihood

Log Integrated Gaussian Likelihood

Description

Compute the log of the integrated Gaussian likelihood, where the model paramaters are integrated out with respect to a normal-inverse gamma prior.

Usage

```
LogIntegratedGaussianLikelihood(suf, prior)
```

Arguments

suf	A GaussianSuf object describing the data.
prior	A NormalInverseGammaPrior describing the prior distribution.

Value

Returns a scalar giving the log integrated likelihood.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
prior <- NormalInverseGammaPrior(10, 2, sqrt(2), 1)
y <- c(7.8949, 9.17438, 8.38808, 5.52521)
suf <- GaussianSuf(y)
loglike <- LogIntegratedGaussianLikelihood(suf, prior)
# -9.73975
```

---

lognormal.prior *Lognormal Prior Distribution*

Description

Specifies a lognormal prior distribution.

Usage

```
LognormalPrior(mu = 0.0, sigma = 1.0, initial.value = NULL)
```

Arguments

mu	mean of the corresponding normal distribution.
sigma	standard deviation of the corresponding normal distribution. WARNING: If something looks strange in your program, look out for SD != Variance errors.
initial.value	Initial value of the variable to be modeled (e.g. in an MCMC algorithm). If NULL then the prior mean will be used.

Details

A lognormal distribution, where log(y) ~ N(mu, sigma). The mean of this distribution is exp(mu + 0.5 * sigma^2), so don't only focus on the mean parameter here.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

https://en.wikipedia.org/wiki/Log-normal_distribution

`markov.prior` *Prior for a Markov chain*

Description

The conjugate prior distribution for the parameters of a homogeneous Markov chain. The rows in the transition probability matrix modeled with independent Dirichlet priors. The distribution of the initial state is modeled with its own independent Dirichlet prior.

Usage

```
MarkovPrior(prior.transition.counts = NULL,
            prior.initial.state.counts = NULL,
            state.space.size = NULL,
            uniform.prior.value = 1)
```

Arguments

`prior.transition.counts`
: A matrix of the same dimension as the transition probability matrix being modeled. Entry (i, j) represents the "prior count" of transitions from state `i` to state `j`.

`prior.initial.state.counts`
: A vector of positive numbers representing prior counts of initial states.

`state.space.size`
: If both prior.transition.counts and prior.initial.state.counts are missing, then they will be filled with an object of dimension state.space.size where all entries are set to `uniform.prior.value`.

`uniform.prior.value`
: The default value to use for entries of `prior.transition.counts` and `prior.initial.state.counts`, when they are not supplied by the user.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

match_data_frame *MatchDataFrame*

Description

Given two data frames with the same data, but with rows and columns in potentially different orders, produce a pair of permutations such that data2[row.permutation, column.permutation] matches data1.

Usage

```
MatchDataFrame(data.to.match, data.to.permute)
```

Arguments

data.to.match	The data frame to be matched.
data.to.permute	The data frame to be permuted.

Value

Returns a list with two elements.

column.permutation
: A vector of indices such that the columns of data2[,column.permutation] match the columns of data1. The matching is based on column names.

row.permutation
: A vector of indices such that the rows of data2[row.permutation,column.permutation] match the rows of data1. The matching is done by converting rows to strings, and matching the strings.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
x1 <- data.frame(larry = rnorm(10), moe = 1:10, curly = rpois(10, 2))
x2 <- x1[c(1:5, 10:6), c(3, 1, 2)]

m <- MatchDataFrame(x1, x2)
x2[m$row.permutation, m$column.permutation] == x1  ## all TRUE
```

mscan *Scan a Matrix*

Description

Quickly scan a matrix from a file.

Usage

```
mscan(fname, nc = 0, header = FALSE, burn = 0, thin = 0, nlines = 0L,
      sep = "", ...)
```

Arguments

fname	The name of the file from which to scan the data.
nc	The number of columns in the matrix to be read. If zero then the number of columns will be determined by the number of columns in the first line of the file.
header	logical indicating whether the file contains a header row.
burn	An integer giving the number of initial lines of the matrix to discard.
thin	An integer. If thin > 1 then keep every thin\'th line. This is useful for reading in very large files of MCMC output, for example.
nlines	If positive, the number of data lines to scan from the data file (e.g. for an MCMC algorithm that is only partway done). Otherwise the entire file will be read.
sep	Field separator in the data file.
...	Extra arguments passed to 'scan'.

Details

This function is similar to read.table, but scanning a matrix of homogeneous data is much faster because there is much less format deduction.

Value

The matrix stored in the data file.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
filename <- file.path(tempdir(), "example.data")
cat("foo bar baz", "1 2 3", "4 5 6", file = filename, sep = "\n")
m <- mscan(filename, header = TRUE)
m
##      foo bar baz
## [1,]   1   2   3
## [2,]   4   5   6
```

`mvn.diagonal.prior` *diagonal MVN prior*

Description

A multivariate normal prior distribution formed by the product of independent normal margins.

Usage

```
MvnDiagonalPrior(mean.vector, sd.vector)
```

Arguments

`mean.vector`	A vector giving the mean of the prior distribution.
`sd.vector`	The standard deviations of the components in the distribution. I.e. the square root of the diagonal of the variance matrix.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

`mvn.independent.sigma.prior`
Independence prior for the MVN

Description

A prior for the parameters of the multivariate normal distribution that assumes Sigma to be a diagonal matrix with elements modeled by independent inverse Gamma priors.

Usage

```
MvnIndependentSigmaPrior(mvn.prior, sd.prior.list)
```

Arguments

mvn.prior	An object of class MvnPrior that is the prior distribution for the multivariate normal mean parameter.
sd.prior.list	A list of SdPrior objects modeling the diagonal elements of the multivariate normal variance matrix. The off-diagonal elements are assumed to be zero.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

---

mvn.prior *Multivariate normal prior*

Description

A multivariate normal prior distribution.

Usage

```
MvnPrior(mean, variance)
```

Arguments

mean	A vector giving the mean of the prior distribution.
variance	A symmetric positive definite matrix giving the variance of the prior distribution.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

normal.inverse.gamma.prior
Normal inverse gamma prior

Description

The NormalInverseGammaPrior is the conjugate prior for the mean and variance of the scalar normal distribution. The model says that

$$\frac{1}{\sigma^2} \sim Gamma(df/2, ss/2)\mu|\sigma \sim N(\mu_0, \sigma^2/\kappa)$$

Usage

```
NormalInverseGammaPrior(mu.guess, mu.guess.weight = .01,
       sigma.guess, sigma.guess.weight = 1, ...)
```

Arguments

mu.guess	The mean of the prior distribution. This is μ_0 in the description above.
mu.guess.weight	The number of observations worth of weight assigned to mu.guess. This is κ in the description above.
sigma.guess	A prior estimate at the value of sigma. This is $\sqrt{ss/df}$.
sigma.guess.weight	The number of observations worth of weight assigned to sigma.guess. This is df.
...	blah

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

`normal.inverse.wishart.prior` *Normal inverse Wishart prior*

Description

The NormalInverseWishartPrior is the conjugate prior for the mean and variance of the multivariate normal distribution. The model says that

$$\Sigma^{-1} \sim Wishart(\nu, S)\mu|\sigma \sim N(\mu_0, \Sigma/\kappa)$$

The $Wishart(S, \nu)$ distribution is parameterized by `S`, the *inverse* of the sum of squares matrix, and the scalar degrees of freedom parameter `nu`.

The distribution is improper if $\nu < dim(S)$.

Usage

```
NormalInverseWishartPrior(mean.guess,
                          mean.guess.weight = .01,
                          variance.guess,
                          variance.guess.weight = nrow(variance.guess) + 1)
```

Arguments

`mean.guess`	The mean of the prior distribution. This is μ_0 in the description above.
`mean.guess.weight`	The number of observations worth of weight assigned to `mean.guess`. This is κ in the description above.
`variance.guess`	A prior estimate at the value of Σ. This is S^{-1}/ν in the notation above.
`variance.guess.weight`	The number of observations worth of weight assigned to `variance.guess`. This is df.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

normal.prior *Normal (scalar Gaussian) prior distribution*

Description

Specifies a scalar Gaussian prior distribution.

Usage

```
NormalPrior(mu, sigma, initial.value = mu, fixed = FALSE)
```

Arguments

mu	The mean of the prior distribution.
sigma	The standard deviation of the prior distribution.
initial.value	The initial value of the parameter being modeled in the MCMC algorithm.
fixed	Should the deviate modeled by this distribution be fixed at its initial value? (Used for debugging in some code. Not universally respected.)

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

pairs.density *Pairs plot for posterior distributions.*

Description

A pairs plot showing the posterior distribution of the given list of Monte Carlo draws. Plots above the diagonal show the posterior distribution on a scale just wide enough to fit the plots. The diagonal shows a marginal density plot, and the subdiagonal shows the distribution with all plots on a common scale.

Usage

```
PairsDensity(draws,
             nlevels = 20,
             lty = NULL,
             color = NULL,
             subset = NULL,
             labels,
             legend.location = "top",
             legend.cex = 1,
             label.cex = 1,
             ...)
```

Arguments

draws	Either a matrix or a list of matrices. If a list is provided then each list element is plotted as a separate set of contours, and all matrices must have the same number of columns (though the number of rows can differ).
nlevels	The number of contour levels to plot.
lty	The line types to use for the different elements in draws.
color	The color to use for different elements in draws.
subset	If draws is a list, then this can be a numerical vector. If draws has names, then subset can be a character vector naming which elements to include. If NULL then all elements of draws are plotted.
labels	If labels is missing and the first element of draws has non-NULL colnames then these will be used to label the pairs plot. If a character vector of length ncol(draws[[1]]) then this character vector will be used in place of the colnames. If NULL then no labels will be used.
legend.location	Either "top", or "right" specifying the location for the legend, or NULL, indicating that no legend is desired. if draws is a matrix or a singleton list then no legend is produced.
legend.cex	Scale factor to use for the legend labels.
label.cex	Scale factor to use for the row and column labels.
...	Extra arguments (graphical parameters), passed to plot, PlotDensityContours, axis, and AddExternalLegend.

Author(s)

Steven L. Scott

See Also

pairs, CompareDensities, CompareManyDensities

Examples

```
## You can see the pairs plot for a single set of draws.
y <- matrix(rnorm(5000, mean = 1:5), ncol = 5, byrow = TRUE)
PairsDensity(y)

## You can also compare two or more sets of draws.
z <- matrix(rnorm(2500, mean = 2:6), ncol = 5, byrow = TRUE)
PairsDensity(list("first set" = y, "second set" = z))
```

`plot.density.contours` *Contour plot of a bivariate density.*

Description

Contour plot of one ore more bivariate densities. This function was originally created to implement PairsDensity, but might be useful on its own.

Usage

```
PlotDensityContours(draws,
                    x.index = 1,
                    y.index = 2,
                    xlim = NULL,
                    ylim = NULL,
                    nlevels = 20,
                    subset = NULL,
                    color = NULL,
                    lty = NULL,
                    axes = TRUE,
                    ...)
```

Arguments

`draws`	Either a matrix or a list of matrices. If a list is provided then each list element is plotted as a separate set of contours, and all matrices must have the same number of columns (though the number of rows can differ).
`x.index`	The index of the parameter to plot on the horizonal axis.
`y.index`	The index of the beta coefficient to plot on the vertical axis.
`xlim`	Limits on the horizontal axis. If NULL then the plot is just wide enough to fit the contours.
`ylim`	Limits on the vertical axis. If NULL then the plot is just tall enough to fit the contours.
`nlevels`	The number of contour levels to plot.

subset	If draws is a list, then this can be a numerical vector. If draws has names, then subset can be a character vector naming which elements to include. If NULL then all elements of draws are plotted.
color	The color to use for different elemetns in draws.
lty	The line types to use for the different elements in draws.
axes	Logical. Should x and y axies be drawn?
...	Extra arguments passed to contour.

Author(s)

Steven L. Scott

See Also

contour, kde2d

Examples

```
## You can see the pairs plot for a single set of draws.
y <- matrix(rnorm(5000, mean = 1:5), ncol = 5, byrow = TRUE)
PlotDensityContours(y, 3, 1)

## You can also compare two or more sets of draws.
z <- matrix(rnorm(2500, mean = 2:6), ncol = 5, byrow = TRUE)
PlotDensityContours(list("first set" = y, "second set" = z), 3, 1)
```

---

plot.dynamic.distribution

Plots the pointwise evolution of a distribution over an index set.

Description

Produces an dynamic distribution plot where gray scale shading is used to show the evolution of a distribution over an index set. This function is particularly useful when the index set is too large to do side-by-side boxplots.

Usage

```
PlotDynamicDistribution(curves,
                        timestamps = NULL,
                        quantile.step=.01,
                        xlim = NULL,
                        xlab = "Time",
                        ylim = range(curves, na.rm = TRUE),
                        ylab = "distribution",
                        add = FALSE,
                        axes = TRUE,
                        ...)
```

Arguments

curves	A matrix where each row represents a curve (e.g. a simulation of a time series from a posterior distribution) and columns represent different points in the index set. For example, a long time series would be a wide matrix.
timestamps	An optional vector of "time stamps" that curves will be plotted against. The length of timestamps must match the number of columns in curves. If timestamps is NULL then the function attempts to extract time stamps from the colnames(curves). If no appropriate time stamps can be found then the positive integers will be used as time stamps.
quantile.step	Each color step in the plot corresponds to this difference in quantiles. Smaller values make prettier plots, but the plots take longer to produce.
xlim	The x limits (x1, x2) of the plot. Note that x1 > x2 is allowed and leads to a "reversed axis".
xlab	Label for the horzontal axis.
ylim	The y limits (y1, y2) of the plot. Note that y1 > y2 is allowed and leads to a "reversed axis".
ylab	Label for the vertical axis.
add	Logical. If true then add the plot to the current plot. Otherwise a fresh plot will be created.
axes	Logical. Should axes be added to the plot?
...	Extra arguments to pass on to plot

Details

The function works by passing many calls to polygon. Each polygon is associated with a quantile level, with darker shading near the median.

Value

This function is called for its side effect, which is to produce a plot on the current graphics device.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
x <- t(matrix(rnorm(1000 * 100, 1:100, 1:100), nrow=100))
## x has 1000 rows, and 100 columns.  Column i is N(i, i^2) noise.

PlotDynamicDistribution(x)
time <- as.Date("2010-01-01", format = "%Y-%m-%d") + (0:99 - 50)*7
PlotDynamicDistribution(x, time)
```

plot.macf *Plots individual autocorrelation functions for many-valued time series*

Description

Produces individual autocorrelation functions for many-valued time series such as those produced by highly multivariate MCMC output. Cross-correlations such as those produced by acf are not shown.

Usage

```
PlotMacf(x, lag.max = 40, gap = 0.5, main = NULL, boxes = TRUE,
         xlab = "lag", ylab = "ACF", type = "h")
```

Arguments

x	matrix or 3-way array of MCMC output (or other time series). The first dimension represents discrete time.
lag.max	maximum lag to use when computing ACF's.
gap	non-negative scalar. gap between plots
main	character. main title for the plot
boxes	logical. Should boxes be drawn around the plots
xlab	character label for horizontal axis.
ylab	character label for vertical axis.
type	type of line plot to show. Defaults to "h". See plot.default for other options.

Value

Called for its side effect

Author(s)

Steven L. Scott

See Also

acf, plot.many.ts.

Examples

```
x <- matrix(rnorm(1000), ncol=10)
PlotMacf(x)
```

`plot.many.ts` *Multiple time series plots*

Description

Plots many time series plots on the same graphical device. Each plot gets its own frame. Scales can be adjusted to see variation in each plot (each plot gets its own scale), or variation between plots (common scale).

Usage

```
PlotManyTs(x, type = "l", gap = 0, boxes = TRUE, truth = NULL,
           thin = 1, labs, same.scale = TRUE, ylim = NULL,
           refline = NULL, color = NULL, ...)
```

Arguments

`x`	Matrix, data frame, list, or 3-dimensional array to be plotted.
`type`	type of line plots to produce. See `plot.default` for other options.
`gap`	Number of lines of space to put between plots.
`boxes`	Logical indicating whether boxes should be drawn around each plot.
`truth`	A vector or matrix of reference values to be added to each plot as a horizontal line. The dimension should match `dim(x)[-1]`
`thin`	Frequency of observations to plot. E.g. `thin=10` means plot every 10'th observation. Thinning can speed things up when plotting large amounts MCMC output.
`labs`	Optional character vector giving the title (e.g. variable name) for each plot. If `labs` is missing then column names or dimnames will be used to label the plots. If `labs` is `NULL` then no labels will be used.
`same.scale`	Logical indicating whether plots should be drawn with a common vertical axis, which is displayed on alternating rows of the plot. If `FALSE` then the vertical axis of each plot is scaled to the range of data in that plot, but no tick marks are displayed.
`ylim`	Scale of the vertical axis. If non-NULL then same.scale is set to `TRUE` and the supplied scale is used for all plots.
`refline`	a vector or scalar value to use as a reference line. This is a supplement to the `truth` argument. It can be useful when comparing true values (used in a simulation), estimated values (e.g. point estimates of parameters) and MCMC output.
`color`	Vector of colors to use in the plots.
`...`	Extra arguments passed to `plot` and `axis`.

Author(s)

Steven L. Scott

See Also

plot.ts (for plotting a small number of time series) plot.macf

Examples

```
x <- matrix(rnorm(1000), ncol = 10)
PlotManyTs(x)
PlotManyTs(x, same = FALSE)
```

---

regression.coefficient.conjugate.prior

Regression Coefficient Conjugate Prior

Description

A conjugate prior for regression coefficients, conditional on residual variance, and sample size.

Usage

```
RegressionCoefficientConjugatePrior(
  mean,
  sample.size,
  additional.prior.precision = numeric(0),
  diagonal.weight = 0)
```

Arguments

mean	The mean of the prior distribution, denoted 'b' below. See Details.
sample.size	The value denoted κ below. This can be interpreted as a number of observations worth of weight to be assigned to mean in the posterior distribution.
additional.prior.precision	A vector of non-negative numbers representing the diagonal matrix Λ^{-1} below. Positive values for additional.prior.precision will ensure the distribution is proper even if the regression model has no data. If all columns of the design matrix have positive variance then additional.prior.precision can safely be set to zero. A zero-length numeric vector is a slightly more efficient equivalent to a vector of all zeros.
diagonal.weight	The weight given to the diagonal when XTX is averaged with its diagonal. The purpose of diagonal.weight is to keep the prior distribution proper even if X is less than full rank. If the design matrix is full rank then diagonal.weight can be set to zero.

Details

A conditional prior for the coefficients (beta) in a linear regression model. The prior is conditional on the residual variance σ^2, the sample size n, and the design matrix X. The prior is

$$\beta|\sigma^2, X \sim N(b, \sigma^2(\Lambda^{-1} + V$$

where

$$V^{-1} = \frac{\kappa}{n}((1-w)X^TX + wdiag(X^TX)).$$

The prior distribution also depends on the cross product matrix XTX and the sample size n, which are not arguments to this function. It is expected that the underlying C++ code will get those quantities elsewhere (presumably from the regression modeled by this prior).

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

`replist` *Repeated Lists of Objects*

Description

Produces repeated copies of an object.

Usage

```
RepList(object, times)
```

Arguments

`object`	The object to be replicated.
`times`	The desired number of replications.

Value

Returns a list containing `times` copies of `object`.

Author(s)

Steven L. Scott

Examples

```
alist <- list(x = "foo", y = 12, z = c(1:3))
three.copies <- RepList(alist, 3)
```

rmvn *Multivariate Normal Simulation*

Description

Simulate draws from the multivariate normal distribution.

Usage

```
rmvn(n = 1, mu, sigma = diag(rep(1., length(mu))))
```

Arguments

n	The desired number of draws.
mu	The mean of the distribution. A vector.
sigma	The variance matrix of the distribution. A matrix.

Details

Note that mu and sigma are the same for all n draws. This function cannot handle separate parameters for each draw the way rnorm and similar functions for scalar random variables can.

Value

If n == 1 the return value is a vector. Otherwise it is a matrix with n rows and length(mu) columns.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
y1 <- rnorm(1, 1:3)
## y1 is a vector
y2 <- rnorm(10, 1:3)
## y2 is a matrix
```

`rvectorfunction` *RVectorFunction*

Description

A wrapper for passing R functions to C++ code.

Usage

```
RVectorFunction(f, ...)
```

Arguments

`f`	A scalar-valued function of a vector-valued argument. The function can depend on other arguments as long as the vector valued argument appears in the first position.
`...`	Optional, named, extra arguments to be passed to f. These arguments are fixed at the time this object is created. For the purpose of evaluating f, these arguments do not update.

Details

The Boom library can handle the output of this function as a C++ function object. Note that evaluating R functions in C is no faster than evaluating them in R, but a wrapper like this is useful for interacting with C and C++ libraries that expect to operate on C and C++ functions.

Value

A list containing the information needed to evaluate the function f in C++ code.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

`scaled.matrix.normal.prior`
Scaled Matrix-Normal Prior

Description

A matrix-normal prior distribution, intended as the conjugate prior for the regression coefficients in a multivariate linear regression.

Usage

```
ScaledMatrixNormalPrior(mean, nu)
```

Arguments

mean	A matrix giving the mean of the distributions
nu	A scale factor affecting the variance.

Details

The matrix normal distribution is a 3-parameter distribution MN(mu, Omega, V), where mu is the mean. A deviate from the distribution is a matrix B, where Cov(B[i, j], B[k, m]) = Omega[i, k] * Sigma[j, m]. If b = Vec(B) is the vector obtained by stacking columns of B, then b is multivariate normal with mean Vec(mu) and covariance matrix

$$\Sigma \otimes Omega$$

(the kronecker product).

This prior distribution assumes the underlying C++ code knows where to find the predictor (X) matrix in the regression, and the residual variance matrix Sigma. The assumed prior distribution is B ~ MN(mu, X'X / nu, Sigma).

Like most other priors in Boom, this function merely encodes information expected by the underlying C++ code, ensuring correct names and formatting.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

---

sd.prior *Prior for a standard deviation or variance*

Description

Specifies an inverse Gamma prior for a variance parameter, but inputs are defined in terms of a standard deviation.

Usage

```
SdPrior(sigma.guess, sample.size = .01, initial.value = sigma.guess,
        fixed = FALSE, upper.limit = Inf)
```

Arguments

sigma.guess	A prior guess at the value of the standard deviation.
sample.size	The weight given to sigma.guess. Interpretable as a prior observation count.
initial.value	The initial value of the paramter in the MCMC algorithm.
fixed	Logical. Some algorithms allow you to fix sigma at a particular value. If TRUE then sigma will remain fixed at initial.value, if supported.
upper.limit	If positive, this is the upper limit on possible values of the standard deviation parameter. Otherwise the upper limit is assumed infinite. Not supported by all MCMC algorithms.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

`sufstat.Rd` *Sufficient Statistics*

Description

Sufficient statistics for various models.

Usage

```
RegressionSuf(X = NULL,
              y = NULL,
              xtx = crossprod(X),
              xty = crossprod(X, y),
              yty = sum(y^2),
              n = length(y),
              xbar = colMeans(X))

GaussianSuf(y)
```

Arguments

`X`	The predictor matrix for a regression problem.
`y`	The data, or the regression response variable.
`xtx`	The cross product of the design matrix. "X transpose X."
`xty`	The cross product of the design matrix with the response vector. "X transpose y."
`yty`	The sum of the squares of the response vector. "y transpose y."
`n`	The sample size.
`xbar`	A vector giving the average of each column in the predictor matrix.

Value

The returned value is a function containing the sufficient statistics for a regression model. Arguments are checked to ensure they have legal values. List names match the names expected by underlying C++ code.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
X <- cbind(1, matrix(rnorm(3 * 100), ncol = 3))
y <- rnorm(100)

## Sufficient statistics can be computed from raw data, if it is
## available.
suf1 <- RegressionSuf(X, y)

## The individual components can also be computed elsewhere, and
## provided as arguments.  If n is very large, this can be a
## substantial coomputational savings.
suf2 <- RegressionSuf(xtx = crossprod(X),
                      xty = crossprod(X, y),
                      yty = sum(y^2),
                      n = 100,
                      xbar = colMeans(X))
```

---

suggest.burn.log.likelihood

Suggest MCMC Burn-in from Log Likelihood

Description

Suggests a burn-in period for an MCMC chain based on the log likelihood values simulated in the final few iterations of the chain.

Usage

```
SuggestBurnLogLikelihood(log.likelihood, fraction = .10, quantile = .9)
```

Arguments

`log.likelihood`	A numeric vector giving the log likelihood values for each MCMC iteration.
`fraction`	The fraction of the chain that should be used to determine the log likelihood lower bound. The default setting looks in the final 25% of the MCMC run. Must be an number less than 1. If `fraction` <= `0` then a 0 burn-in is returned.
`quantile`	The quantile of the values in the final fraction that must be exceeded before the burn-in period is declared over.

Details

Looks at the last `fraction` of the `log.likelihood` sequence and finds a specified quantile to use as a threshold. Returns the first iteration where `log.likelihood` exceeds this threshold.

Value

Returns a suggested number of iterations to discard. This can be 0 if `fraction == 0`, which is viewed as a signal that no burn-in is desired.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

`thin` *Thin the rows of a matrix*

Description

Systematic sampling of every `thin`'th row of a matrix or vector. Useful for culling MCMC output or denoising a plot.

Usage

```
thin(x, thin)
```

Arguments

`x`	The array to be thinned. The first dimension is the one sampled over.
`thin`	The frequency of observations to keep. With `thin=10` you will keep every 10th observation.

Value

The thinned vector, matrix, or array is returned.

Author(s)

Steven L. Scott

Examples

```
x <- rnorm(100)
thin(x, 10)
# returns a 10 vector

y <- matrix(rnorm(200), ncol=2)
thin(y, 10)
# returns a 10 by 2 matrix
```

`thin.matrix` *Thin a Matrix*

Description

Return discard all but every k'th row of a matrix.

Usage

```
ThinMatrix(mat, thin)
```

Arguments

`mat`	The matrix to be thinned.
`thin`	The distance between kepts lines from mat. The larger the number the fewer lines are kept.

Details

The bigger the value of `thin` the more thinning that gets done. For example, `thin = 10` will keep every 10 lines from `mat`.

Value

The matrix mat, after discarding all but every `thin` lines.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
m <- matrix(1:100, ncol = 2)
ThinMatrix(m, thin = 10)
##      [,1] [,2]
## [1,]   10   60
## [2,]   20   70
## [3,]   30   80
## [4,]   40   90
## [5,]   50  100
```

TimeSeriesBoxplot *Time Series Boxplots*

Description

Creates a series of boxplots showing the evolution of a distribution over time.

Usage

```
TimeSeriesBoxplot(x, time, ylim = NULL, add = FALSE, ...)
```

Arguments

x	A matrix where each row represents a curve (e.g. a simulation of a time series from a posterior distribution) and columns represent time. A long time series would be a wide matrix.
time	A vector of class Date with lenght matching the number of columns in x.
ylim	limits for the y axis.
add	logical, if TRUE then add boxplots to current plot.
...	Extra arguments to pass on to boxplot

Value

Called for its side effect, which is to produce a plot on the current graphics device.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
x <- t(matrix(rnorm(1000 * 100, 1:100, 1:100), nrow=100))
## x has 1000 rows, and 100 columns.  Column i is N(i, i^2) noise.
time <- as.Date("2010-01-01", format = "%Y-%m-%d") + (0:99 - 50)*7
TimeSeriesBoxplot(x, time)
```

ToString *Convert to Character String*

Description

Convert an object to a character string, suitable for including in error messages.

Usage

```
ToString(object, ...)
```

Arguments

object	An object to be printed to a string.
...	Extra arguments passed to print.

Value

A string (a character vector of length 1) containing the printed value of the object.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Examples

```
m <- matrix(1:6, ncol = 2)
printed.matrix <- ToString(m)

y <- c(1, 2, 3, 3, 3, 3, 3, 3)
tab <- table(y)
printed.table <- ToString(tab)
```

traceproduct *Trace of the Product of Two Matrices*

Description

Returns the trace of the product of two matrices.

Usage

```
TraceProduct(A, B, b.is.symmetric = FALSE)
```

Arguments

A	The first matrix in the product.
B	The second matrix in the product.
b.is.symmetric	Logical. A TRUE value indicates that B is a symmetric matrix. A slight computational savings is possible if B is symmetric.

Value

Returns a number equivalent to sum(diag(A %*% B)).

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

---

uniform.prior *Uniform prior distribution*

Description

Specifies a uniform prior distribution on a real-valued scalar parameter.

Usage

```
UniformPrior(lo = 0, hi = 1, initial.value = NULL)
```

Arguments

lo	The lower limit of support.
hi	The upper limit of support.
initial.value	The initial value of the parameter in question to use in the MCMC algorithm. If NULL then the mean (lo + hi)/2 is used.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

References

Gelman, Carlin, Stern, Rubin (2003), "Bayesian Data Analysis", Chapman and Hall.

wishart *Wishart Distribution*

Description

Density and random generation for the Wishart distribution.

Usage

```
dWishart(W, Sigma, nu, logscale = FALSE)
rWishart(nu, scale.matrix, inverse = FALSE)
```

Arguments

W	Argument (random variable) for the Wishart density. A symmetric positive definite matrix.
Sigma	Scale or "variance" parameter of the Wishart distribution. See the "details" section below.
nu	The "degrees of freedom" parameter of the Wishart distribution. The distribution is only defined for nu >= nrow(W) -1.
logscale	Logical. If TRUE then the density is returned on the log scale. Otherwise the density is returned on the density scale.
scale.matrix	For the Wishart distribution the scale.matrix parameter means the same thing as the Sigma parameter in dWishart. It is the variance parameter of the generating multivariate normal distribution. If simulating from the inverse Wishart, scale.matrix is the INVERSE of the "sum of squares" matrix portion of the multivariate normal sufficient statistics.
inverse	Logical. If TRUE then simulate from the inverse Wishart distribution. If FALSE then simulate from the Wishart distribution.

Details

If nu is an integer then a $W(\Sigma, \nu)$ random variable can be thought of as the sum of nu outer products: $y_i y_i^T$, where y_i is a zero-mean multivariate normal with variance matrix Sigma.

The Wishart distribution is

$$\frac{|W|^{\frac{\nu-p-1}{2}} \exp(-tr(\Sigma^{-1}W)/2)}{2^{\frac{\nu p}{2}} |\Sigma|^{\frac{\nu}{2}} \Gamma_p(\nu/2)}$$

where p == nrow(W) and $\Gamma_p(\nu)$ is the multivariate gamma function (see lmgamma).

Value

`dWishart` returns the density of the Wishart distribution. It is not vectorized, so only one random variable (matrix) can be evaluated at a time.

`rWishart` returns one or more draws from the Wishart or inverse Wishart distributions. If `n > 0` the result is a 3-way array. Unlike the `rWishart` function from the stats package, the first index corresponds to draws. This is in keeping with the convention of other models from the Boom package.

Author(s)

Steven L. Scott <steve.the.bayesian@gmail.com>

Index