

# Package ‘mcmcplots’

March 19, 2015

**Type** Package

**Title** Create Plots from MCMC Output

**Version** 0.4.2

**Date** 2015-03-15

**Maintainer** S. McKay Curtis <s.mckay.curtis@gmail.com>

**Depends** coda (>= 0.17.1)

**Imports** sfsmisc, colorspace, denstrip

**Description** Functions for convenient plotting and viewing of MCMC output.

**License** GPL (>= 2)

**LazyLoad** yes

**LazyData** yes

**NeedsCompilation** no

**Author** S. McKay Curtis [aut, cre],  
Ilya Goldin [ctb],  
Evangelos Evangelou [ctb],  
'sumtxt' from GitHub [ctb]

**Repository** CRAN

**Date/Publication** 2015-03-19 06:47:34

## R topics documented:

mcmcplots-package . . . . .	2
.graypr . . . . .	5
.html.begin . . . . .	6
.html.end . . . . .	7
.html.img . . . . .	7
.to.greek . . . . .	8
as.mcmc.bugs . . . . .	9
as.mcmc.rjags . . . . .	10
autplot1 . . . . .	11
caterplot . . . . .	12

caterpoints . . . . .	16
convert.mcmc.list . . . . .	17
corplot . . . . .	17
denoverplot . . . . .	19
denoverplot1 . . . . .	20
denplot . . . . .	22
mcmcplot . . . . .	24
mcmcplot1 . . . . .	26
mcmcplotsPalette . . . . .	27
parcorplot . . . . .	29
parms2plot . . . . .	30
rmeanplot . . . . .	32
rmeanplot1 . . . . .	33
traplot . . . . .	34
traplot1 . . . . .	36

<b>Index</b>	<b>38</b>
--------------	-----------

---

mcmcplots-package	<i>Plots for MCMC Output</i>
-------------------	------------------------------

---

## Description

Viewing diagnostics plots for MCMC output is often difficult when a Bayesian model has a large number of parameters. Fitting all density or trace plots in a single plotting window is not possible when the number of parameters is large. One common solution is to create one plot window at a time and prompt the user before creating each plot. However, clicking through plot windows can be tedious and slow.

This package attempts to address these problems by providing a function (`mcmcplot`) that produces common MCMC diagnostic plots in an html file that can be viewed from a web browser. When viewed in a web browser, hundreds of MCMC plots can be viewed efficiently by scrolling through the output as if it were any typical web page.

Also, `mcmcplot` and other functions in this package – `denplot`, `traplot`, `caterplot` – have arguments that facilitate selecting subsets of parameters to plot. For example, specifying the `parms` argument to be "beta" will prompt any of the previous functions to create plots for parameters with names that start with "beta", such as "beta[1]", "beta[2]", and so on. Additionally, specifying the `random` option in any of these functions will produce plots for a random subset of parameters in the model.

This package also contains other plotting functions which can be useful in developing and debugging MCMC software written in R. The `denoverplot` function creates overlaying density plots of all common parameters from two different MCMC simulations. This function can be useful when debugging MCMC software. MCMC software is sometimes written in stages, where the first stage of development involves writing an MCMC sampler in a high-level programming language like R or WinBUGS. If the program is too slow to be practical for most data sets, then the second stage of development involves rewriting the MCMC sampler in a low-level language like C or Fortran. If overlaying density plots show slight differences, then the new, low-level code likely has bugs.

The `corplot` function (see also `parcorplot`) creates a "heat plot" of a correlation matrix. This function can be useful in deciding on a blocking structure for an MCMC algorithm, because highly correlated parameters can be sampled in a single block of an MCMC algorithm to improve efficiency.

## Details

Package: mcmcplots  
Type: Package  
Version: 0.4  
Date: 2012-10-07  
License: GPL (>= 2)  
LazyLoad: yes

## Author(s)

S. McKay Curtis with contributions from Ilya Goldin

Maintainer: S. McKay Curtis <s.mckay.curtis@gmail.com>

This research was supported in part by Grant R01 AG 029672 from the National Institute on Aging, Paul K. Crane, PI.

## References

None.

## See Also

**coda**

## Examples

```
## Not run:
## mcmcplots functions work on bugs objects too
library(R2WinBUGS)
example("openbugs", "R2WinBUGS")
## from the help file for openbugs:
schools.sim <- bugs(data, inits, parameters, model.file,
                   n.chains = 3, n.iter = 5000,
                   program = "openbugs", working.directory = NULL)
caterplot(schools.sim, "theta")
trapplot(schools.sim, "theta")
denplot(schools.sim, "theta")
mcmcplot(schools.sim)

## End(Not run)
```

```

## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

## Use mcmcplot to plot
## the fake MCMC output
## Not run:
mcmcplot(fakemcmc)
mcmcplot(fakemcmc, "gamma")
mcmcplot(fakemcmc, regex="alpha\\[[12]]")
mcmcplot(fakemcmc, "gamma", "alpha\\[[12]]")
mcmcplot(fakemcmc, random=2)
mcmcplot(fakemcmc, random=c(2, 3))

## End(Not run)

## Use traplot to create
## trace plots of fake MCMC data
traplot(fakemcmc)
traplot(fakemcmc, "gamma")
traplot(fakemcmc, "gamma", "alpha\\[[12]]$") # all gamma and alpha[1] and alpha[2]

## Use denplot to create
## density plots of fake MCMC data
denplot(fakemcmc)
denplot(fakemcmc, "gamma")
denplot(fakemcmc, "gamma", "alpha\\[[12]]$") # all gamma and alpha[1] and alpha[2]

## Use caterplot to create
## caterpillar plots of fake MCMC data
par(mfrow=c(2,2))
caterplot(fakemcmc, "alpha", collapse=FALSE)
caterplot(fakemcmc, "gamma", collapse=FALSE)
caterplot(fakemcmc, "alpha", labels.loc="axis", col="blue")
caterplot(fakemcmc, "gamma", labels.loc="above", col="red")

## Use caterplot to create
## caterpillar plots of fake MCMC data
## with density strips
caterplot(fakemcmc, "alpha", collapse=FALSE, denstrip=TRUE)
caterplot(fakemcmc, "gamma", collapse=FALSE, denstrip=TRUE)
caterplot(fakemcmc, "alpha", labels.loc="axis", col="blue", denstrip=TRUE)
caterplot(fakemcmc, "gamma", labels.loc="above", col="red", denstrip=TRUE)

## Overlay caterplots
caterplot(fakemcmc, "alpha", collapse=TRUE)
caterplot(fakemcmc, "gamma", collapse=TRUE, add=TRUE, cat.shift=-0.3)

```

```

## Use denoverplot to create overlaying density plots
## of all parameters in fake MCMC data
fakemcmc2 <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means, each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))
denoverplot(fakemcmc, fakemcmc2)

## Use corplot to create a "heat plot" of a
## correlation matrix of the fake MCMC draws
corplot(cor(as.matrix(fakemcmc)), cex.axis=0.75) ## not exciting
Rho1 <- outer(1:10, 1:10, function(i, j) 0.5^(abs(i-j)))
Rho2 <- outer(1:5, 1:5, function(i, j) 0.25^(i!=j))
dat1 <- t(apply(matrix(rnorm(10*1000), 1000, 10), 1,
  function(z, Rho1) crossprod(Rho1, z), Rho1))
dat2 <- t(apply(matrix(rnorm(5*1000), 1000, 5), 1,
  function(z, Rho2) crossprod(Rho2,z), Rho2))
colnames(dat1) <- paste("theta[", 1:10, "]", sep="")
colnames(dat2) <- paste("alpha[", 1:5, "]", sep="")
dat <- cbind(dat1, dat2)
parcorplot(dat, "theta", col=gray(31:0/31), cex.axis=0.75) ## just theta parameters
parcorplot(dat, col=heat.colors(31), cex.axis=0.75)
parcorplot(dat, col=topo.colors(31), cex.axis=0.75)
parcorplot(dat, col=terrain.colors(31), cex.axis=0.75)
parcorplot(dat, col=cm.colors(31), cex.axis=0.75)

```

`.graypr`*Create a Gray Plotting Region*

## Description

Colors the plotting region gray and plots light-gray gridlines. This function is intended for internal use only.

## Usage

```

.graypr(x.axis = TRUE, y.axis = TRUE,
x.major = TRUE, y.major = TRUE,
x.minor = TRUE, y.minor=TRUE,
x.malty = 1, y.malty = 1,
x.milty = 1, y.milty = 1)

```

## Arguments

<code>x.axis</code>	if TRUE an x axis will be plotted.
<code>y.axis</code>	if TRUE a y axis will be plotted.
<code>x.major</code>	if TRUE, major gridlines on the x axis will be plotted.
<code>y.major</code>	if TRUE, major gridlines on the y axis will be plotted.

<code>x.minor</code>	if TRUE, minor gridlines on the x axis will be plotted.
<code>y.minor</code>	if TRUE, minor gridlines on the y axis will be plotted.
<code>x.malty</code>	line type to be used on the x-axis, major gridlines.
<code>y.malty</code>	line type to be used on the y-axis, major gridlines.
<code>x.milty</code>	line type to be used on the x-axis, minor gridlines.
<code>y.milty</code>	line type to be used on the y-axis, minor gridlines.

**Author(s)**

S. McKay Curtis

**See Also**

`rect`

---

`.html.begin`

*Initialize an html file for use in the `mcmcplot` function.*

---

**Description**

Initializes an html file to capture output from the `mcmcplot` function. Intended for internal use only.

**Usage**

```
.html.begin(outdir = tempdir(), filename = "index", extension = "html", title, cssfile)
```

**Arguments**

<code>outdir</code>	file path for the output directory.
<code>filename</code>	name of the html file.
<code>extension</code>	name of the file extension.
<code>title</code>	title for the html file.
<code>cssfile</code>	css file name.

**Value**

String containing the path to the initialized html file.

**Author(s)**

Ilya Goldin

**See Also**

[.html.end](#), [.html.img](#), [mcmcplot](#)

### Examples

```
## See examples for function mcmcplot
```

---

<code>.html.end</code>	<i>Complete the html file for use in mcmcplot</i>
------------------------	---

---

### Description

Completes the html file that was generated by the mcmcplot function. Intended for internal use only.

### Usage

```
.html.end(file)
```

### Arguments

file            a string with a path to the html file.

### Author(s)

Ilya Goldin

### See Also

[.html.begin](#), [.html.img](#), [mcmcplot](#)

### Examples

```
## See examples for function mcmcplot
```

---

<code>.html.img</code>	<i>Insert an image into an html file</i>
------------------------	--

---

### Description

Embeds an image into an html file generated by the mcmcplot function. Intended for internal use only.

### Usage

```
.html.img(file, ...)
```

### Arguments

file            string containing the h  
...             string containing additional html attributes for the image.

**Author(s)**

Ilya Goldin

**See Also**[.html.begin](#), [.html.end](#), [mcmcplot](#)**Examples**

```
## See examples for function mcmcplot
```

---

<code>.to.greek</code>	<i>Parse strings containing names of greek letters.</i>
------------------------	---

---

**Description**

Convert character vector with greek letters to an expression suitable for plotting greek symbols in titles and labels.

**Usage**

```
.to.greek(instr)
```

**Arguments**

`instr` a character vector

**Value**

Parsed representation of the character vector.

**Author(s)**

Ilya Goldin

**See Also**[parse](#)**Examples**

```
## Not run: .to.greek(c("alpha", "beta", "mybeta", "YourDelta"))
```



---

`as.mcmc.bugs`*Convert a bugs Object to an mcmc or mcmc.list Object*

---

**Description**

Converts a bugs object to an mcmc object.

**Usage**

```
## S3 method for class 'bugs'  
as.mcmc(x, ...)
```

**Arguments**

<code>x</code>	bugs object
<code>...</code>	unused

**Value**

If `x` contains multiple chains, the function returns an `mcmc.list` object. Otherwise, the function returns an `mcmc` object.

**Author(s)**

S. McKay Curtis

**See Also**

bugs in **R2WinBUGS**

**Examples**

```
## Not run:  
## Data object "schools.sim" generated from the examples  
## in the bugs function of the R2WinBUGS package.  
outmcmc <- as.mcmc(schools.sim)  
  
## Gelman Rubin diagnostics  
:gelman.diag(outmcmc)  
:mcmc.plot(outmcmc)  
  
## End(Not run)
```

---

`as.mcmc.rjags`*Convert an rjags Object to an mcmc or mcmc.list Object.*

---

**Description**

Converts an rjags object to an mcmc object.

**Usage**

```
## S3 method for class 'rjags'  
as.mcmc(x, ...)
```

**Arguments**

<code>x</code>	rjags object
<code>...</code>	unused

**Details**

If `x` contains multiple chains, the function returns an `mcmc.list` object. Otherwise, the function returns an `mcmc` object.

**Value**

An `mcmc.list` or `mcmc` object.

**Author(s)**

S. McKay Curtis

**See Also**

`coda` package.

**Examples**

```
## None ##
```

---

`autplot1`*Autocorrelation Plot of MCMC Output*

---

**Description**

Creates an autocorrelation or partial autocorrelation plot of MCMC output.

**Usage**

```
autplot1(x, chain = 1, lag.max = NULL, partial = FALSE,  
col = mcmcplotsPalette(1), style = c("gray", "plain"),  
ylim = NULL, ...)
```

**Arguments**

<code>x</code>	an <code>mcmc.list</code> object with a single variable.
<code>chain</code>	the number of the parallel chain for plotting. The default is to use the first parallel chain.
<code>lag.max</code>	passed as an argument to the autocorrelation function <code>acf</code> .
<code>partial</code>	logical indicating whether partial autocorrelation should be plotted.
<code>col</code>	color of the bars in the plot.
<code>style</code>	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
<code>ylim</code>	limits for the y-axis.
<code>...</code>	further arguments passed to the plotting function.

**Details**

None.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis (adapted from Martyn Plummer's `autcorr.plot` code in the **coda** package)

**References**

None.

**See Also**

[acf](#), [autocorr.plot](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <-
  coda::as.mcmc.list(
    lapply(1:3,
      function(i)
        coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
                          nrow=nr, dimnames=list(NULL,pnames))))))

autplot1(fakemcmc[, "alpha[1]", drop=FALSE])
autplot1(fakemcmc[, "alpha[1]", drop=FALSE], chain=2, style="plain")
autplot1(fakemcmc[, "alpha[1]", drop=FALSE], partial=TRUE)
```

---

caterplot

*Caterpillar Plots of MCMC Output*


---

**Description**

Creates plots of credible intervals for parameters from an MCMC simulation. Because these types of plots have been called "caterpillar" plots by other Bayesian software (like WinBUGS), this function is called *caterplot*, where the "cat" is pronounced as in caterpillar and not as in the word "cater".)

**Usage**

```
caterplot(mcmcout, parms = NULL, regex = NULL, random = NULL,
  leaf.marker = "[\\[_]", quantiles = list(), collapse = TRUE,
  reorder = collapse, denstrip = FALSE, add = FALSE, labels = NULL,
  labels.loc = "axis", las = NULL, cex.labels = NULL, greek = FALSE,
  horizontal=TRUE, val.lim = NULL, lab.lim = NULL, lwd = c(1, 2),
  pch = 16, eps = 0.1, width = NULL, col = NULL, cat.shift=0,
  style=c("gray", "plain"), ...)
```

**Arguments**

mcmcout	a matrix, bugs, mcmc, or mcmc.list object. All objects will be coerced to mcmc.list.
parms	a vector of character strings that identifies which variables in mcmcout should be plotted. If parms and regex are both NULL, all parameter will be plotted.
regex	a vector of character strings with regular expressions that identify which variables in mcmcout should be plotted.

random	integer specifying how many parameters from each group will be randomly selected for plotting. This argument is useful when <code>mcmcout</code> has a large number of parameters (e.g., from a hierarchical model). If <code>NULL</code> , all parameters will be plotted.
leaf.marker	a regular expression with a character class that marks the beginning of the “leaf” portion of a parameter name. The default character class includes <code>[</code> and <code>_</code>
quantiles	list with two elements <code>outer</code> and <code>inner</code> . The <code>outer</code> element of the list should contain the quantiles of the posterior draws that will be plotted as the longer, thinner line. The <code>inner</code> element of the list should contain the quantiles of posterior draws that will be plotted as the shorter, thicker line. If missing, the default is to use <code>list(outer=c(0.025, 0.975), inner=c(0.16, 0.84))</code> , which corresponds to 95% and 68% credible intervals.
collapse	if <code>TRUE</code> , all parallel chains are collapsed into one chain before plotting. If <code>FALSE</code> , parallel chains are plotted nearly on top of each other with colors as specified in <code>col</code> .
reorder	if <code>TRUE</code> , caterpillars will be ordered according to their medians. This option only works when <code>collapse=TRUE</code> .
denstrip	if <code>TRUE</code> , then density strips will be plotted rather than quantile line plots.
add	if <code>TRUE</code> , output will be added to the existing plot.
labels	labels for the individual “caterpillars.” If <code>NULL</code> , parameter names in <code>mcmcout</code> are used.
labels.loc	if ‘axis,’ then parameter labels (the names of the parameters) will be plotted on the axis. If ‘above,’ then variable names will be plotted above the means for each ‘caterplot.’ If any other value, no names will be plotted.
las	controls the rotation of the labels on the label axis. See documentation for <code>par</code> for more information.
cex.labels	character expansion factor for the plot labels. If names of parameters will not be plotted, this argument is ignored.
greek	if <code>TRUE</code> , the names of greek letters in the labels will be displayed as greek characters on the plot.
horizontal	logical indicating whether intervals should be plotted parallel to the x-axis (so <i>horizontal</i> lines) or parallel to the y axis.
val.lim	a vector containing the upper and lower limits for the “value” axis (which is the x axis if <code>horizontal=TRUE</code> ). If <code>NULL</code> , the minimum and maximum values found by outer quantiles are used.
lab.lim	a vector containing the upper and lower limits for the “label” axis (which is the y axis if <code>horizontal=TRUE</code> ). If <code>NULL</code> , limits are automatically selected to contain all the labels of the plotted intervals.
col	a single value or a vector of values specifying the colors to be used in plotting. Default is <code>mcmcplotsPalette(nchains)</code>
lwd	a vector of length 2 of line weights used for plotting the inner and outer caterpillars.
pch	plot character to use in plotting the medians of the intervals.

eps	controls the spacing between parallel caterpillars when collapse=FALSE
width	width of the density strips.
cat.shift	if greater than 0, "caterpillars" are translated up (left) by the amount cat.shift. If less than 0, "caterpillars" are translated down (right) by the amount cat.shift.
style	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
...	further arguments passed to the plotting function.

### Details

The caterplot function uses the internal function parms2plot to match the strings in the parms argument to the names of the variables in mcmcout. Quantiles, as specified in the quantiles argument, are computed for the posterior draws of each variable returned by the call to parms2plot. The quantiles are then used to create plots of the posterior intervals of each matched variable. Medians are also plotted. If the option denstrip is set to TRUE, then density strips are plotted instead of quantile lines. (See Jackson, 2008.)

This function produces a plot similar to the plots produced by the coefplot function in the R package **arm** and the caterpillar plots in the WinBUGS software.

### Value

Invisibly returns a character vector with the names of the parameters that were plotted. This can be useful when the option random is specified and not all of the parameters are plotted. See [caterpoints](#) for an example of how to use the return value.

### Note

None.

### Author(s)

S. McKay Curtis

### References

Jackson, C. H. (2008) "Displaying uncertainty with shading". *The American Statistician*, 62(4):340-347.

### See Also

[caterpoints](#), [mcmcplot](#), [denstrip](#), [parms2plot](#)

### Examples

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[1,", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
```

```

    lapply(1:3,
           function(i)
             coda::mcmc(matrix(rnorm(nc*nr, rep(means, each=nr)),
                               nrow=nr, dimnames=list(NULL,pnames))))))

## caterplot plots of the fake MCMC output
par(mfrow=c(2,2))
caterplot(fakemcmc, "alpha", collapse=FALSE)
caterplot(fakemcmc, "gamma", collapse=FALSE)
caterplot(fakemcmc, "alpha", labels.loc="axis", greek=TRUE, col="blue")
caterplot(fakemcmc, "gamma", labels.loc="above", greek=TRUE, col="red")

caterplot(fakemcmc, "alpha", collapse=FALSE, denstrip=TRUE)
caterplot(fakemcmc, "gamma", collapse=FALSE, denstrip=TRUE)
caterplot(fakemcmc, "alpha", labels.loc="axis", col="blue", denstrip=TRUE)
caterplot(fakemcmc, "gamma", labels.loc="above", col="red", denstrip=TRUE)

caterplot(fakemcmc, "alpha", collapse=FALSE, style="plain")
caterplot(fakemcmc, "gamma", collapse=FALSE, style="plain")
caterplot(fakemcmc, "alpha", labels.loc="axis")
caterplot(fakemcmc, "gamma", labels.loc="above")

caterplot(fakemcmc, "alpha", horizontal=FALSE)
caterplot(fakemcmc, horizontal=FALSE)
caterpoints(rnorm(10, 21, 2), horizontal=FALSE, pch="x", col="red")
caterplot(fakemcmc, horizontal=FALSE, denstrip=TRUE, col="blue", pch=NA)
caterplot(fakemcmc, horizontal=FALSE, col="red", pch=19, add=TRUE)
caterplot(fakemcmc, denstrip=TRUE, col="blue", pch=NA)
caterplot(fakemcmc, col="purple", pch=19, add=TRUE)

## Overlay caterplots
caterplot(fakemcmc, "alpha", collapse=TRUE)
caterplot(fakemcmc, "gamma", collapse=TRUE, add=TRUE, cat.shift=-0.3)

## What happens with NULL varnames?
coda::varnames(fakemcmc) <- NULL
caterplot(fakemcmc)
caterplot(fakemcmc, collapse=FALSE)

## Not run:
## caterplot works on bugs objects too:
library(R2WinBUGS)
example("openbugs", "R2WinBUGS")
## from the help file for openbugs:
schools.sim <- bugs(data, inits, parameters, model.file,
                   n.chains = 3, n.iter = 5000,
                   program = "openbugs", working.directory = NULL)
caterplot(schools.sim, "theta")

## End(Not run)

```

---

caterpoints	<i>Points on a "caterplot"</i>
-------------	--------------------------------

---

**Description**

Adds points to a caterplot.

**Usage**

```
caterpoints(x, parnames, horizontal = TRUE, ...)
```

**Arguments**

x	vector of points to add to a caterplot.
parnames	an optional vector of parameter names. If specified, x must have a names attribute. The argument parnames will be used to subset the vector x for plotting, as in x[parnames].
horizontal	logical value that should match the argument of the same name in the original call to caterplot.
...	further arguments passed to the function points.

**Author(s)**

S. McKay Curtis

**See Also**

[caterplot](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pname <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means, each=nr)),
      nrow=nr, dimnames=list(NULL,pname))))))
posterior.medians <- apply(do.call("rbind", fakemcmc), 2, median)

## caterplot plots of the fake MCMC output
par(mfrow=c(2,2))
caterplot(fakemcmc, "alpha", collapse=FALSE)
caterpoints(runif(5, 10, 20), pch="x", col="red")
caterplot(fakemcmc, "alpha", horizontal=FALSE)
caterpoints(runif(5, 10, 20), horizontal=FALSE, pch="x", col="red")
```



```
parms <- caterplot(fakemcmc, random=3) # keep the names of plotted parameters
caterpoints(posterior.medians[parms], pch="x", col="red")
```

---

convert.mcmc.list      *Convert an object to mcmc.list object*

---

### Description

Attempts to convert any object to an mcmc.list object. Intended for internal use only.

### Usage

```
convert.mcmc.list(x)
```

### Arguments

x                      an object.

### Value

An object of class mcmc.list.

### Author(s)

S. McKay Curtis

---

corplot                      *Plot a Correlation Matrix*

---

### Description

Creates an image plot of a correlation matrix where colors of different shades represent differing levels of correlation.

### Usage

```
corplot(mat, col = mcmcplotsPalette(11, "sequential"), outline = TRUE,
greek = FALSE, legend.scale = 0.75, mar=c(5, 4, 1, 1) + 0.1, ...)
```

**Arguments**

mat	correlation matrix.
col	colors to be used in the plot.
outline	logical indicating whether outlines of image squares should be drawn.
greek	if TRUE, the names of greek letters in the labels will be displayed as greek characters on the plot.
legend.scale	scales the height of the legend with respect to the height of the plot. Default is 0.75 which makes the legend 3 quarters as tall as the plot.
mar	graphical parameter mar. See documentation for par.
...	further arguments passed to the plotting function.

**Details**

One possible use of this function is to plot the correlation between posterior draws of an MCMC run. Patterns in the plot can aid in constructing a more efficient blocking structure for an MCMC algorithm, where highly correlated parameters should be placed in the same MCMC update block. None.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

image

**Examples**

```
Rho <- matrix(c(
  1.00, 0.35, -0.65, -0.66, 0.46, 0.42,
  0.35, 1.00, -0.69, -0.64, 0.40, -0.06,
  -0.65, -0.69, 1.00, 0.70, -0.57, -0.11,
  -0.66, -0.64, 0.70, 1.00, -0.15, -0.10,
  0.46, 0.40, -0.57, -0.15, 1.00, 0.18,
  0.42, -0.06, -0.11, -0.10, 0.18, 1.00), 6, 6)
dimnames(Rho) <- list(paste("rho[", 1:6, "]", sep=""), paste("rho[", 1:6, "]", sep=""))
corplot(Rho)
corplot(Rho, greek=TRUE)
```

**Description**

Determines which parameters are in common from two different MCMC simulations and plots overlaying density estimates of the parameters in common.

**Usage**

```
denoverplot(mcmc1, mcmc2, parms = NULL, regex = NULL, random = NULL,
ci = NULL, auto.layout = TRUE, legend = TRUE,
mar = c(2.0, 2.0, 1.5, 0.25) + 0.1, col = mcmcplotsPalette(2),
lty = 1, plot.title = NULL, main = NULL, greek = FALSE,
style = c("gray", "plain"), ...)
```

**Arguments**

mcmc1	object that can be coerced to an mcmc object.
mcmc2	object that can be coerced to an mcmc object.
parms	character vector specifying which subsets of parameters to plot. If NULL and regex=NULL, denoverplot will plot all common parameters mcmc1 and mcmc2. See documentation for <a href="#">parms2plot</a> for more information.
regex	character vector of regular expressions denoting groups of parameters to plot.
random	integer specifying how many parameters from each group will be randomly selected for plotting. This argument is useful when mcmc2 has a large number of parameters (e.g., from a hierarchical model). If NULL, mcmcplot will plot all parameters.
ci	if non NULL, plots $(100*ci)\%$ credible interval limits on the density plots. The default (ci=NULL) is not to plot the intervals.
auto.layout	logical specifying whether the <code>multi.fig</code> function from the <b>sfsmisc</b> package should be used to construct the plotting region.
legend	if TRUE an extra plot in the plotting region is used as a legend.
mar	argument passed to <code>multi.fig</code> if <code>auto.layout=TRUE</code>
col	colors for plotting the densities.
lty	line types for plotting densities. Argument is recycled to be of length 2.
plot.title	title to put in the outer margin. Default is no title.
main	character vector of titles to put over each individual plot. If NULL, then the names of the parameters are used.
greek	if TRUE, the names of greek letters in the labels will be displayed as greek characters on the plot.
style	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
...	additional arguments passed to the denoverplot function.

**Details**

This function can be used in debugging MCMC code by comparing distributions of parameters from the development MCMC code and a reference MCMC simulation.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

[denplot](#), [parms2plot](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means, each=nr)),
                                  nrow=nr, dimnames=list(NULL,pnames))))))
fakemcmc2 <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means, each=nr)),
                                  nrow=nr, dimnames=list(NULL,pnames))))))

## Plot the fake MCMC output
denoverplot(fakemcmc, fakemcmc2)
denoverplot(fakemcmc, fakemcmc2, style="plain",
            col=mcmcplotsPalette(3, type="grayscale"),
            ci=0.95, greek=TRUE)
denoverplot(fakemcmc, fakemcmc2,
            plot.title="Comparison of densities of fake data")
denoverplot(fakemcmc, fakemcmc2,
            plot.title="Comparison of densities of fake data", greek=TRUE)
```

---

denoverplot1

*Plot Overlaying Densities*

---

**Description**

Creates a plot containing overlaying kernel density estimates from different MCMC simulations. This function is used in the `denoverplot` function to produce plots of overlaying densities for parameters in common from two different MCMC simulations.

**Usage**

```
denoverplot1(..., ci = NULL, col = NULL, lty = 1, xlim = NULL,
ylim = NULL, xlab = "", ylab = "Density", main = NULL, style = c("gray",
"plain"), gpar = NULL)
```

**Arguments**

...	one or more vectors or a list containing one or more vectors to be plotted.
ci	if non NULL, plots $(100*ci)\%$ credible interval limits on the density plots. The default (NULL) is not to plot the intervals.
col	one or more colors for the densities. Default is <code>mcmcplotsPalette(n)</code> , where <code>n</code> is the number of elements in the ... argument.
lty	types of lines to plot.
xlim	limits for the x axis.
ylim	limits for the y axis.
xlab	label for the x axis.
ylab	label for the y axis.
main	main title for plot.
style	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
gpar	a list of additional graphical parameters to be passed to the plotting function. See help for <code>par</code> .

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis with contributions from Ilya Goldin

**See Also**

[denoverplot](#), [denplot](#), [traplot1](#)

**Examples**

```
denoverplot1(rnorm(1000), rnorm(1000))
denoverplot1(rnorm(1000, 0.0, 1.0), rnorm(1000, 0.1, 1.0),
             style="plain", col=mcmcplotsPalette(2, type="grayscale"),
             ci=0.95)
denoverplot1(list(rgamma(1000, 1, 1), rgamma(1000, 1, 1)))
```

denplot

*Density Plots for MCMC Parameters on a Single Plot***Description**

Creates a plot of densities for specified parameters from an MCMC simulation in a single plot or plots densities for those parameters as indicated by the `parms` and `regex`.

**Usage**

```
denplot(mcmcout, parms = NULL, regex = NULL, random = NULL,
leaf.marker="[\\[_]", ci = NULL, xlim = NULL,
ylim = NULL, auto.layout = TRUE, mar=c(2.0, 2.0, 1.5, 0.25) + 0.1, col =
NULL, lty = 1, xlab = "", ylab = "", plot.title = NULL, main = NULL,
greek = FALSE, collapse = FALSE, style=c("gray", "plain"), ...)
```

**Arguments**

<code>mcmcout</code>	an object that can be coerced to an <code>mcmc</code> or <code>mcmc.list</code> object
<code>parms</code>	a vector of character strings that identifies which variables in <code>mcmcout</code> should be plotted. If <code>parms</code> and <code>regex</code> are both <code>NULL</code> , all parameters will be plotted.
<code>regex</code>	a vector of character strings with regular expressions that identify which variables in <code>mcmcout</code> should be plotted.
<code>random</code>	an integer indicating the maximum number of parameters to randomly select for plotting from each group of parameters as specified by the <code>parms</code> argument.
<code>leaf.marker</code>	a regular expression with a character class that marks the beginning of the “leaf” portion of a parameter name. The default character class includes <code>[</code> and <code>_</code>
<code>ci</code>	if non <code>NULL</code> , plots $(100*ci)\%$ credible interval limits on the density plots. The default ( <code>NULL</code> ) is not to plot the intervals.
<code>xlim</code>	limits of the x-axis.
<code>ylim</code>	limits of the y-axis.
<code>auto.layout</code>	if <code>TRUE</code> , <code>denplot</code> automatically creates a plot layout using <code>multi.fig</code> from the <b>sfsmisc</b> package.
<code>mar</code>	argument passed to <code>multi.fig</code> if <code>auto.layout=TRUE</code>
<code>col</code>	colors to be used in plotting the densities.
<code>lty</code>	line types to be used in plotting.
<code>xlab</code>	label for the x-axis.
<code>ylab</code>	label for the y-axis.
<code>plot.title</code>	title to put in the outer margin. Default is no title.
<code>main</code>	character vector of titles for each individual plot. If <code>NULL</code> , then the names of the parameters are used.

<code>greek</code>	if TRUE, the names of greek letters in the labels will be displayed as greek characters on the plot.
<code>collapse</code>	if TRUE, all parallel chains are collapsed into one chain before plotting. If FALSE, parallel chains are plotted with colors as specified in <code>col</code> .
<code>style</code>	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
<code>...</code>	further arguments to be passed to the plotting function.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

[mcmcplot](#), [parms2plot](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

## Plot densities of the fake MCMC output
denplot(fakemcmc)
denplot(fakemcmc, style="plain")
denplot(fakemcmc, collapse=TRUE, greek=TRUE, ci=0.95)
denplot(fakemcmc, xlim=range(unlist(fakemcmc)),
  plot.title="Density plots of fake data. Yawn.")
denplot(fakemcmc, "gamma")
denplot(fakemcmc, "gamma", "alpha\\[[12]]$") # all gamma and alpha[1] and alpha[2]

## What happens with NULL varnames?
coda::varnames(fakemcmc) <- NULL
denplot(fakemcmc)

## Not run:
## denplot works on bugs objects too
library(R2WinBUGS)
example("openbugs", "R2WinBUGS")
## from the help file for openbugs:
schools.sim <- bugs(data, inits, parameters, model.file,
  n.chains = 3, n.iter = 5000,
```

```

                                program = "openbugs", working.directory = NULL)
denplot(schools.sim, "theta")

## End(Not run)

```

---

mcmcplot

*Diagnostics Plots for MCMC in HTML format*


---

## Description

Creates an HTML file that displays diagnostic plots (trace, density, autocorrelation) from an MCMC simulation. When the number of parameters in an MCMC simulation is large, viewing all plots in a web browser is much easier than clicking through R graph windows.

## Usage

```

mcmcplot(mcmcout, parms = NULL, regex = NULL, random = NULL,
leaf.marker = "[\\[_]", dir = tempdir(), filename = "MCMCoutput",
extension = "html", title = NULL,
heading = title, col = NULL, lty = 1,
xlim = NULL, ylim = NULL,
style=c("gray", "plain"), greek = FALSE)

```

## Arguments

mcmcout	posterior draws. This argument will be coerced to an mcmc object.
parms	character vector specifying subsets of parameters to plot. If parms=NULL and regex=NULL, mcmcplot will plot all parameters.
regex	character vector of regular expressions denoting groups of parameters to plot.
random	integer specifying how many parameters from each group will be randomly selected for plotting. This argument is useful when mcmcout has a large number of parameters (e.g., from a hierarchical model). If NULL, mcmcplot will plot all parameters.
leaf.marker	a regular expression with a character class that marks the beginning of the “leaf” portion of a parameter name. The default character class includes [ and _
dir	string containing the directory where the plots and the main html file will be stored.
filename	string containing the name of the main html file which will contain code to display each plot produced by mcmcplot.
extension	string containing the extension to be used for the html file.
title	string containing the title to be included in the html file. Default is to use the name of object passed as the mcmcout argument prepended with the string "MCMC Output: ".
heading	string containing the heading to be used for the html file. Default is to use the title.



col	vector of colors. This will determine the colors that will be used to plot each chain in the traceplots and density plots. Default is <code>mcmcplotsPalette(nchains)</code> .
lty	vector of line types. This will determine the line types that will be used to plot each chain in the traceplots and density plots. If missing, <code>mcmcplot</code> will use 1 for all line types.
xlim	limits for the x axis of the density plot.
ylim	limits for the y axis of the density plot.
style	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
greek	if TRUE, the names of greek letters in the labels will be displayed as greek characters on the plot.

### Details

The `mcmcplot` function generates an html file that contains diagnostics plots – trace plots, autocorrelation plots, and density plots – for parameters from an MCMC simulation. When an MCMC simulation contains a large number of parameters, it is no longer convenient to view plots in a small graph window. Viewing the plots in a web browser gives the user the ability to scroll through and examine a large number of plots in a more convenient manner.

See documentation for [parms2plot](#) for more information on how to “smartly” select parameters to plot using the `parms`, `regex`, and `random` arguments.

### Value

Invisibly returns a string containing the path to filename.

### Author(s)

S. McKay Curtis and Ilya Goldin

### See Also

[parms2plot](#), plotting functions in the `coda` package.

### Examples

```
## Not run:
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

## Use mcmcplot to plot
## the fake MCMC output
mcmcplot(fakemcmc)
```

```

mcmcplot(fakemcmc, greek=TRUE)
mcmcplot(fakemcmc, xlim=range(fakemcmc)) # put the densities on the same scale
mcmcplot(fakemcmc, "gamma")
mcmcplot(fakemcmc, regex="alpha\\[[12]\\]", style="plain")
mcmcplot(fakemcmc, "gamma", regex="alpha\\[[12]\\]")
mcmcplot(fakemcmc, random=2)
mcmcplot(fakemcmc, random=c(2, 3))

## What happens with NULL varnames?
coda::varnames(fakemcmc) <- NULL
mcmcplot(fakemcmc)

## mcmcplot works on bugs objects too
library(R2WinBUGS)
example("openbugs", "R2WinBUGS")
## from the help file for openbugs:
schools.sim <- bugs(data, inits, parameters, model.file,
                    n.chains = 3, n.iter = 5000,
                    program = "openbugs", working.directory = NULL)
mcmcplot(schools.sim)

## End(Not run)

```

---

mcmcplot1

*MCMC Diagnostics Plots for one Model Parameter*


---

## Description

Creates a graph window containing three different plots—a trace plot, a density plot, and an autocorrelation plot—for one parameter in an MCMC run. This function is used by `mcmcplot` to construct an html file of MCMC diagnostics. This function is intended for internal use only.

## Usage

```

mcmcplot1(x, col = mcmcplotsPalette(n), lty = 1, xlim = NULL, ylim =
NULL, style = c("gray", "plain"), greek = FALSE)

```

## Arguments

<code>x</code>	an mcmc object with a single variable.
<code>col</code>	colors for plotting each parallel chain. The default is <code>seq(nchains)+1</code> where <code>nchains</code> is the number of parallel chains in <code>mcmcout</code> . If there is only one parallel chain, then the default is 1.
<code>lty</code>	line types for plotting each parallel chain. The default is 1 for all parallel chains.
<code>xlim</code>	limits for the x axis of the density plot.
<code>ylim</code>	limits for the y axis of the density plot.
<code>style</code>	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.

`greek` if TRUE, the names of greek letters in the labels will be displayed as greek characters on the plot.

**Value**

Creates a plot.

**Note**

Only the first parallel chain is used to create the autocorrelation plot. This function is used by `mcmcplot` to create html output for all the parameters of an MCMC simulation.

**Author(s)**

S. McKay Curtis

**References**

No references.

**See Also**

[mcmcplot](#)

**Examples**

```
## Create fake MCMC output
fakemcmc <- coda::as.mcmc.list(coda::mcmc(sapply(1:5, function(dum) rnorm(1000))))
coda::varnames(fakemcmc) <- c("gamma[1,1]", "gamma[1,2]", "gamma[1,3]", "sigma[1]", "sigma[2]")

mcmcplot1(fakemcmc[, "sigma[1]", drop=FALSE])
mcmcplot1(fakemcmc[, "gamma[1,3]", drop=FALSE], style="plain")
```

---

mcmcplotsPalette

*Color Palette for the mcmcplots Package*

---

**Description**

Creates a color palette for plotting functions in the **mcmcplots** package using functions in the **colorspace** package.

**Usage**

```
mcmcplotsPalette(n, type = c("rainbow", "sequential",
"grayscale"), seq = NULL)
```

**Arguments**

n	number of colors
type	denotes the type of color palette to create.
seq	deprecated

**Value**

A color palette of n colors.

**Author(s)**

S. McKay Curtis

**References**

Zeileis, A., Hornik, K. and Murrell, P. (2009) "Escaping RGBland: Selecting colors for statistical graphs." *Computational Statistics & Data Analysis*, 53, 3259–3270.

**See Also**

rainbow\_hcl, sequential\_hcl

**Examples**

```
colorpie <- function(n, type="rainbow") pie(rep(1, n), col=mcmcplotsPalette(n, type=type))
colorpie(1)
colorpie(8)
colorpie(4, type="sequential")
colorpie(4, type="grayscale")

## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

denplot(fakemcmc)
denplot(fakemcmc, style="plain", col=mcmcplotsPalette(3, type="sequential"))
denplot(fakemcmc, style="plain", col=mcmcplotsPalette(3, type="grayscale"))
```

---

`parcorplot`*Correlation Plot for MCMC Draws of Model Parameters*

---

**Description**

Creates an image plot of posterior correlations between model parameters from an MCMC simulation.

**Usage**

```
parcorplot(mcmcout, parms = NULL, regex = NULL, random = NULL, col = gray(11:0/11), ...)
```

**Arguments**

<code>mcmcout</code>	posterior draws. This argument will be coerced to an <code>mcmc</code> object.
<code>parms</code>	character vector specifying which subsets of parameters to plot. If <code>NULL</code> and <code>regex=NULL</code> , <code>mcmcplot</code> will plot all parameters. Regular expressions are used to strip all numbers and punctuation out of the parameter names to find the parameters that match the character strings in <code>parms</code> .
<code>regex</code>	character vector of regular expressions denoting groups of parameters to plot.
<code>random</code>	integer specifying how many parameters from each group will be randomly selected for plotting. This argument is useful when <code>mcmcout</code> has a large number of parameters (e.g., from a hierarchical model). If <code>NULL</code> , <code>mcmcplot</code> will plot all parameters.
<code>col</code>	colors to be used in the plot.
<code>...</code>	further arguments that are passed to the <code>corplot</code> function.

**Details**

The `parcorplot` is a wrapper function to `corplot` that allows the use of arguments `parms`, `regex`, and `random` to conveniently select parameters from an MCMC simulation to plot with `corplot`.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

[corplot](#), [parms2plot](#)

## Examples

```
Rho1 <- outer(1:10, 1:10, function(i, j) 0.5^(abs(i-j)))
Rho2 <- outer(1:5, 1:5, function(i, j) 0.25^(i!=j))
dat1 <- t(apply(matrix(rnorm(10*1000)), 1000, 10), 1, function(z, Rho1) t(Rho1)%*%z, Rho1))
dat2 <- t(apply(matrix(rnorm(5*1000)), 1000, 5), 1, function(z, Rho2) t(Rho2)%*%z, Rho2))
colnames(dat1) <- paste("theta[", 1:10, "]", sep="")
colnames(dat2) <- paste("alpha[", 1:5, "]", sep="")
dat <- cbind(dat1, dat2)
parcorplot(dat, "theta", col=gray(31:0/31), cex.axis=0.75)
parcorplot(dat, col=heat.colors(31), cex.axis=0.75)
parcorplot(dat, col=topo.colors(31), cex.axis=0.75)
parcorplot(dat, col=terrain.colors(31), cex.axis=0.75)
parcorplot(dat, col=cm.colors(31), cex.axis=0.75)
```

---

parms2plot

*Matches groups of parameters to plot in MCMC diagnostics plots.*

---

## Description

Utility function that finds the parameter names to plot in the `mcmcplot` function. Intended for internal use only.

## Usage

```
parms2plot(parnames, parms, regex, random, leaf.marker = "[\\[_]", do.unlist = TRUE)
```

## Arguments

<code>parnames</code>	parameter names from an MCMC run
<code>parms</code>	partial parameter names that will be used to determine which subset of <code>parnames</code> will be plotted.
<code>regex</code>	a vector of character strings containing regular expressions to match parameter names in the <code>mcmc</code> object.
<code>random</code>	an integer or <code>NULL</code> . If an integer is specified, the function will select only random number of plots from each parameter group for plotting. If <code>NULL</code> , all parameter names from groups specified in <code>parms</code> will be plotted.
<code>leaf.marker</code>	a regular expression with a character class that marks the beginning of the “leaf” portion of a parameter name. The default character class includes <code>[</code> and <code>_</code>
<code>do.unlist</code>	a logical indicating whether the function should return the vector of parameter names or a list of parameter names according to parameter “groupings” (so parameters can be accessed according to their “stems”). This option was added in order to improve the functionality of <code>mcmcplot</code> function.

## Details

The function `parms2plot` is used internally by most plotting functions in the **mcmcplots** package. The function's purpose is to allow users to conveniently specify groups of parameters to be used in plots of MCMC output.

`parms2plot` relies on using regular expressions to find “stems” and “leaves” in parameter names and to create groups of parameters. For example, the parameter `beta[10]` has stem `beta` and leaf `[10]`, and this naming convention indicates that the parameter `beta[10]` is part of a larger collection of `beta` parameters.

`parms2plot` uses a “leaf marker” specified by the `leaf.marker` argument to determine the end of the parameter stem and the beginning of its leaf. The default leaf marker is an open left bracket “[” or an “\_” as specified by a regular expression character class.

Creating plots of specific groupings of parameters is possible by specifying parameter stems in the `parms` argument. For example, calling the function `traplot(mcmcout, parms="beta")` will create a single plot window of trace plots for parameters `beta[1]`, `...`, `beta[10]`.

At first glance the leaf-marker concept might seem like overkill. For example, to plot parameters `mu[1]`, `...`, `mu[10]` why not simply use a string matching function to match “mu” in the parameter names? The answer is that other parameter names might also match “mu” but may not be part of the grouping `mu[1]`, `...`, `mu[10]`. A model with parameter name `mu.gamma` would match the string “mu” but is not part of the parameter grouping `mu[1]`, `...`, `mu[10]`. `parms2plot` avoids this “greedy” matching by requiring an explicit declaration of a leaf marker.

`parms2plot` also allows the user to specify regular expressions for more direct control over the groups of parameters that are plotted. Regular expressions are specified via the `regex` argument. When `parms`, `regex`, and `random` are `NULL`, `parms2plot` will return all parameter names.

The `random` option is useful when an MCMC simulation contains a large number of parameters in a group, e.g. in a hierarchical model with one or more parameter per observation in the data set. In such settings, it is not feasible to create or examine plots for all parameters in a model. The `random` argument allows the user to specify a maximum number of plots to create for each parameter grouping. If a parameter grouping exceeds the number specified in `random`, then a number of parameters (as specified in `random`) will be randomly selected for plotting. If `random` is a vector, then each element of `random` corresponds to a parameter grouping specified in `parms` and `regex`. If specified, the `random` argument is recycled to be the same length as `length(parms) + length(regex)`. Values of `NA` in `random` denote parameter groupings where all parameters in the group will be plotted.

## Value

A character vector with parameter names.

## Author(s)

S. McKay Curtis with contributions from Ilya Goldin

## See Also

[mcmcplot](#), [caterplot](#), [traplot](#), [denplot](#)

**Examples**

```

prm <- c(paste("gamma[", 1:30, "]", sep=""), paste("alpha[", 1:20, "]", sep=""))

parms2plot(prm, NULL, NULL, NULL)      # returns all
parms2plot(prm, NULL, NULL, 5)        # returns 5 randomly from each group
parms2plot(prm, NULL, NULL, c(5, 10)) # 5 from gamma, 10 from alpha
parms2plot(prm, NULL, NULL, c(10, NA)) # 10 from gamma, all from alpha
parms2plot(prm, "alpha", NULL, NULL)  # all alphas
parms2plot(prm, "gamma", NULL, NULL)  # all gammas
parms2plot(prm, NULL, "alpha\\[1[[:digit:]]\\]", NULL) # alpha[10]-alpha[19]
parms2plot(prm, "gamma", "alpha\\[1[[:digit:]]\\]", NULL) # all gamma and alpha[10]-alpha[19]

```

rmeanplot

*Running Mean Plots of Multiple Parameters***Description**

This function produces running mean plots from an MCMC simulation on a single plot for all parameters (by default) or those parameters indicated by the `parms` argument.

**Usage**

```

rmeanplot(mcmcout, parms = NULL, regex = NULL, random = NULL,
leaf.marker = "[\\[_]", ylim = NULL, auto.layout = TRUE,
mar = c(2, 2, 1.5, 0.25) + 0.1, col = NULL, lty = 1,
plot.title = NULL, main = NULL, greek = FALSE,
style = c("gray", "plain"), ...)

```

**Arguments**

<code>mcmcout</code>	an object that can be coerced to an <code>mcmc</code> or <code>mcmc.list</code> object
<code>parms</code>	character vector specifying which subsets of parameters to plot. If <code>NULL</code> , <code>mcmcplot</code> will plot all parameters. Regular expressions are used to strip all numbers and punctuation out of the parameter names to find the parameters that match the character strings in <code>parms</code> .
<code>regex</code>	character vector of regular expressions denoting groups of parameters to plot.
<code>random</code>	an integer indicating the maximum number of parameters to randomly select for plotting from each group of parameters as specified by the <code>parms</code> argument.
<code>leaf.marker</code>	a regular expression with a character class that marks the beginning of the “leaf” portion of a parameter name. The default character class includes <code>[</code> and <code>_</code>
<code>ylim</code>	limits for the y-axis.
<code>auto.layout</code>	automatically creates a plot layout using <code>multi.fig</code> if <code>TRUE</code> .
<code>mar</code>	argument passed to <code>multi.fig</code> if <code>auto.layout=TRUE</code>
<code>col</code>	colors to be used in plotting the densities. Default is <code>mcmcplotsPalette(nchains)</code> .
<code>lty</code>	line types to be used in plotting.



plot.title	title to put in the outer margin. Default is no title.
main	main title for the plots. Default is to use parameter names.
greek	if TRUE, the names of greek letters in the labels will be displayed as greek characters on the plot.
style	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
...	further arguments passed to the plotting function.

**Value**

Creates a plot.

**Author(s)**

Evangelos Evangelou

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

## Plot traces of the fake MCMC output
rmeanplot(fakemcmc)
rmeanplot(fakemcmc, style="plain")
rmeanplot(fakemcmc, "gamma", greek=TRUE)
```

---

rmeanplot1

*Running Mean Plot for a Single Parameter in MCMC Output*


---

**Description**

Creates a trace plot of a running mean for one parameter in an MCMC simulation. This function is called by the mcmcplot function, and is intended for internal use only.

**Usage**

```
rmeanplot1(x, col = NULL, lty = 1, style = c("gray", "plain"), ...)
```

**Arguments**

<code>x</code>	an <code>mcmc.list</code> object with a single variable.
<code>col</code>	one or more colors for the trace lines. Default is <code>mcmcplotsPalette(nchains)</code> .
<code>lty</code>	one or more line types for the trace lines.
<code>style</code>	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
<code>...</code>	further arguments passed to the plotting function.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

[mcmcplot1](#), [denoverplot1](#), [autplot1](#), [traplot1](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

rmeanplot(fakemcmc[, "alpha[5]", drop=FALSE])
rmeanplot(fakemcmc[, "alpha[5]", drop=FALSE, style="plain")
```

---

traplot

*Traceplots of Multiple Parameters.*

---

**Description**

This function produces trace plots from an MCMC simulation on a single plot for all parameters (by default) or those parameters indicated by the `parms` argument.

**Usage**

```
traplot(mcmcout, parms = NULL, regex = NULL, random = NULL,
  leaf.marker="[\\[_]", ylim = NULL, auto.layout = TRUE,
  mar = c(2.0, 2.0, 1.5, 0.25) + 0.1, col =
  NULL, lty = 1, plot.title = NULL, main = NULL,
  greek = FALSE, style = c("gray", "plain"), ...)
```

**Arguments**

<code>mcmc.out</code>	an object that can be coerced to an <code>mcmc</code> or <code>mcmc.list</code> object
<code>parms</code>	character vector specifying which subsets of parameters to plot. If <code>NULL</code> , <code>mcmcplot</code> will plot all parameters. Regular expressions are used to strip all numbers and punctuation out of the parameter names to find the parameters that match the character strings in <code>parms</code> .
<code>regex</code>	character vector of regular expressions denoting groups of parameters to plot.
<code>random</code>	an integer indicating the maximum number of parameters to randomly select for plotting from each group of parameters as specified by the <code>parms</code> argument.
<code>leaf.marker</code>	a regular expression with a character class that marks the beginning of the “leaf” portion of a parameter name. The default character class includes <code>[</code> and <code>_</code>
<code>ylim</code>	limits for the y-axis.
<code>auto.layout</code>	automatically creates a plot layout using <code>multi.fig</code> if <code>TRUE</code> .
<code>mar</code>	argument passed to <code>multi.fig</code> if <code>auto.layout=TRUE</code>
<code>col</code>	colors to be used in plotting the densities. Default is <code>mcmcplotsPalette(nchains)</code> .
<code>lty</code>	line types to be used in plotting.
<code>plot.title</code>	title to put in the outer margin. Default is no title.
<code>main</code>	main title for the plots. Default is to use parameter names.
<code>greek</code>	if <code>TRUE</code> , the names of greek letters in the labels will be displayed as greek characters on the plot.
<code>style</code>	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
<code>...</code>	further arguments passed to the plotting function.

**Value**

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

[denplot](#), [parms2plot](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
```

```

nrow=nr, dimnames=list(NULL,pnames))))))

## Plot traces of the fake MCMC output
traplot(fakemcmc)
traplot(fakemcmc, style="plain")
traplot(fakemcmc, "gamma", greek=TRUE)

## What happens with NULL varnames?
coda::varnames(fakemcmc) <- NULL
traplot(fakemcmc)

## Not run:
## traplot works on bugs objects too
library(R2WinBUGS)
example("openbugs", "R2WinBUGS")
## from the help file for openbugs:
schools.sim <- bugs(data, inits, parameters, model.file,
                   n.chains = 3, n.iter = 5000,
                   program = "openbugs", working.directory = NULL)
traplot(schools.sim, "theta")

## End(Not run)

```

---

traplot1

*Trace Plot for a Single Parameter in MCMC Output*


---

### Description

Creates a trace plot for one parameter in an MCMC simulation. This function is called by the `mcmcplot` function, and is intended for internal use only.

### Usage

```
traplot1(x, col = NULL, lty = 1, style = c("gray", "plain"), ...)
```

### Arguments

<code>x</code>	an <code>mcmc.list</code> object with a single variable.
<code>col</code>	one or more colors for the trace lines. Default is <code>mcmcplotsPalette(nchains)</code> .
<code>lty</code>	one or more line types for the trace lines.
<code>style</code>	if "gray", then the plotting region is printed with a gray background, otherwise the default plotting region is used.
<code>...</code>	further arguments passed to the plotting function.

### Value

Creates a plot.

**Author(s)**

S. McKay Curtis

**See Also**

[mcmcplot1](#), [denoverplot1](#), [autplot1](#)

**Examples**

```
## Create fake MCMC output
nc <- 10; nr <- 1000
pnames <- c(paste("alpha[", 1:5, "]", sep=""), paste("gamma[", 1:5, "]", sep=""))
means <- rpois(10, 20)
fakemcmc <- coda::as.mcmc.list(
  lapply(1:3,
    function(i) coda::mcmc(matrix(rnorm(nc*nr, rep(means,each=nr)),
      nrow=nr, dimnames=list(NULL,pnames))))))

traplot(fakemcmc[, "alpha[5]", drop=FALSE])
traplot(fakemcmc[, "alpha[5]", drop=FALSE], style="plain")
```

# Index

- \*Topic **aplot**
  - .graypr, 5
  - caterpoints, 16
- \*Topic **color**
  - mcmcplotsPalette, 27
- \*Topic **hplot**
  - autplot1, 11
  - caterplot, 12
  - corplot, 17
  - denoverplot, 19
  - denoverplot1, 20
  - denplot, 22
  - mcmcplot, 24
  - mcmcplot1, 26
  - parcorplot, 29
  - rmeanplot, 32
  - rmeanplot1, 33
  - traplot, 34
  - traplot1, 36
- \*Topic **manip**
  - as.mcmc.bugs, 9
  - as.mcmc.rjags, 10
- \*Topic **package**
  - mcmcplots-package, 2
- \*Topic **utilities**
  - .html.begin, 6
  - .html.end, 7
  - .html.img, 7
  - .to.greek, 8
  - convert.mcmc.list, 17
  - parms2plot, 30
- .graypr, 5
- .html.begin, 6, 7, 8
- .html.end, 6, 7, 8
- .html.img, 6, 7, 7
- .to.greek, 8
- acf, 11
- as.mcmc.bugs, 9
- as.mcmc.rjags, 10
- autocorr.plot, 11
- autplot1, 11, 34, 37
- caterplot, 12, 16, 31
- caterpoints, 14, 16
- convert.mcmc.list, 17
- corplot, 17, 29
- denoverplot, 19, 21
- denoverplot1, 20, 34, 37
- denplot, 20, 21, 22, 31, 35
- denstrip, 14
- mcmcplot, 6–8, 14, 23, 24, 27, 31
- mcmcplot1, 26, 34, 37
- mcmcplots (mcmcplots-package), 2
- mcmcplots-package, 2
- mcmcplotsPalette, 27
- parcorplot, 29
- parms2plot, 14, 19, 20, 23, 25, 29, 30, 35
- parse, 8
- rmeanplot, 32
- rmeanplot1, 33
- traplot, 31, 34
- traplot1, 21, 34, 36